universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Control of Levitated Nanoparticles: Single Particle Thermodynamics"

verfasst von / submitted by

Markus Rademacher, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna, 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066876
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Physik
Betreut von / Supervisor:	Dr. Nikolai Kiesel

Abstract

The physics of stochastic thermodynamics has applications ranging from biology over material science to information theory. A prominent example is the Jarzynski equality relating far-from-equilibrium mechanical behaviour of a system to a thermodynamic equilibrium free-energy difference. This has for instance been used to describe the folding properties of nucleic acids and proteins.
Recently, the theoretical study of fast temperature changes, i.e., thermal driving, generalised the Jarzynski equality beyond a purely mechanical control. Despite the importance of processes like heat engine cycles that use both, thermal and mechanical control, we are lacking experimental studies of fluctuation theorems that also take thermal driving into account.
The central goal of this thesis is to fill this gap and assess the William-Searles-Evans equality and the generalized Jarzynski equality experimentally. To achieve this we use levitated nanoparticles as a test bed that enables fast control over the mechanical potential and an effective thermal bath, which is implemented via feedback control.
Crucial for this study was the extension of an existing experiment on a hollow-core photonic crystal fibre optical trap: We implemented programmable time-dependent control of the central experimental parameters and calibrated its influence on the detection system and data analysis. For evaluating the large amount of data collected for this type of experiment, we develop multi-core parallelised programs for quick data processing.
In this Master thesis we successfully demonstrate that the William-Searles-Evans and the generalized Jarzynski fluctuation theorem hold for thermal and mechanical driving far from equilibrium, at a speed up to two orders of magnitude beyond the validity of the quasi-static approximation.

Acknowledgements

I would like to thank my supervisor Nikolai Kiesel for the support, encouragement and critique throughout the course of this master thesis project. He enabled me to be one of the first members of his thermo group and facilitated an exchange of my works with peers at all sorts of different occasions.
Right next in line of persons to which I would like to express my gratitude is Markus Aspelmeyer, as he allowed me to work in his offices and laboratories in the previous years. He did not only offer these resources but he also enabled me to also be part of his group and therefore give me the means to have an environment, where science can flourish in discussions with like-minded people, who as a bonus are experts acknowledged in the community for their field.
In addition to that I would like to thank Michael Konopik of the group of Eric Lutz for helping in tweaking the computer simulation's numeric methods.
I would like to thank the trappers subgroup inside the Aspelmeyer group for never leaving a question of mine unanswered and for the introduction to the afternoon office meetings: Lorenzo Magrini, Uros Delic, David Grass, Manuel Reisenbauer, Tobias Wenzl, Mario Ciampini and Maxime Debiossac. I especially wish to thank David and Maxime; David for letting me use and modify his setup and navigating around the square kilometres of labview code; Maxime for keeping up the positive team spirit even when something was breaking, and for his delicious British lemon tarte.
I want to thank Ramon Moghadas Nia, Andy (Zhenghao) Ding, Klemens Winkler and Joshua Slater for never letting it get boring in our cozy office, which by now definitely has too much computer screen real estate for such a small room (yes I am thinking about you, Andy).
Further I would like to thank Hendrik Ulbricht for hosting my Erasmus exchange in his laboratories in sunny Southampton, UK, and thereby widening my horizon on experimental nano physics. My Erasmus stay will always have a special place, thanks to the Ulbricht group: Muddassar Rashid, Ashley Setter, George Winstone, Chris Timberlake, Jamie Vovrosh, Marko Toroš and David Hempston. Special thanks go to Ashley and George; Ashley for introducing me to the PEP 8 python convention and enabling me to be part of the software library used in this very thesis; George for all the fun weekends in the office and for bouncing (crazy) ideas, which led to my involvement in the surface experiment.
Lastly but most importantly I want to thank my family and friends. I would not have made it without my parents' unconditional love. In addition my girlfriend Eva Kilian deserves the highest of thanks for always putting up with me, supporting me and being understandable when it occasionally came to me spending more hours a week in the lab than at home. Timon Salar Gutleb and Jakob Michl also earn a place for the many distractions in our shared student life and engaging in discussions about philosophy and politics.

Contents

1. Introductory Remarks

1.1. History and State of the Art

Starting in the 19$^{\text{th}}$ century the first steps were taken by physicists to develop a comprehensive theory, which in today's world is known as classical thermodynamics [1]. The world back then necessitated the need for such a physical theory framework, because engineers were lacking a proper theoretical description of the proposed machines, like in publication [2] in (1699), which were utilizing not-yet-understood thermodynamic processes at their core. In particular people were looking for ways to achieve better mechanical engine performances and efficiencies [3, 4]. In the initial days of thermodynamics a set of basic relations were obtained straight from observations of an experiment, which encompassed a gas-filled cylinder with a movable top-wall, also known as a piston [5–7]. These equations stemming from gas-compression experiments are nowadays referred to as the first and second law of classical thermodynamics [8] and can be seen in equations (1.1) and (1.2).

$$dU = \delta W + \delta Q \tag{1.1}$$

The first law of thermodynamics in equation (1.1) basically represents a way to comprehend the composition of the different energy types making up the change of the intrinsic energy when we do a change of states of the system. Thus, dU in relation (1.1) quantifies the overall change of inner energy of the system, δQ stands for heat exchanged between the system and the environment and δW designates the work done during the state change of the system of interest.
The inequality formulation of the second law of thermodynamics as it is written in equation (1.2) provides a statement about the entropy production for the case of an isolated system.

$$dS \geq 0 \tag{1.2}$$

dS in equation (1.2) denotes the change of entropy of the system and reveals one of the characteristic strongholds of classical thermodynamics, namely that the change in entropy over time must only be positive when we deploy a change to the system.
These laws relate to experiments conducted in the 19$^{\text{th}}$ century which were exclusively performed with systems that sufficed the canonical condition. Therefore we have to be careful with the universality of the first and moreover the second law of thermodynamics, equation (1.1) and (1.2). The canonical condition [9, 10] can be described as having several thousand particles, like in a gas, which are collectively manipulated to do a change. In addition to that the system, for instance the gas, always remains in equilibrium with the

surrounding environment while a parameter of the system is changed. The statement about the entropy change of a system given by the second law in equation (1.2) has been formulated for systems of large size. These systems have a large ensemble of particles and perform a change in equilibrium with the surrounding environment. Additionally one can quickly make out that due to these restrictions of the physical system, the theoretical explanation only describes and calculates the average collective behaviour. So rather than taking into account the total information of each and every individual particle in the entire set of particles undergoing the change, we just get the mean of the performed parameter change on the system. So we are essentially missing information, because we are not looking at the phase space trajectories of any particle and are therefore for example not monitoring the inner energy change an individual particle undergoes while we do a controlled change of the system of interest.
The arrival of statistical mechanics in the years around about 1900 marked also the beginning of a new chapter for equilibrium thermodynamics because the physical descriptions started to explain how a manipulation of the system probabilistically affects the change of energy [11]. In fact it was a novel approach to describe a physical system interacting with a heat bath back then, because it comprehended the theory of micro-states, which accounts for all the distinct levels of energy a many particle system can acquire. Therefore it was possible to look at the likelihood of having the entire ensemble of particles in a certain micro-state after the system's evolution governed by a parameter change was finished rather than to just have the mean behaviour of the system of interest. We can express the individual probabilities p_i making up the probability distribution of a specified set of particles as seen in equation (1.3).

$$p_i = e^{-\beta(E_i - F)} \qquad \text{with} \qquad \beta = \frac{1}{k_B T} \tag{1.3}$$

The exponential relation (1.3) gives us the insight to express the probability of finding our physical ensemble in a certain energy level E_i by taking the difference between the energy of the state and the Helmholtz free energy F of the system and multiply it by the Boltzmann factor β. In this representation the free energy F is used as a replacement to the partition function.
Dealing with non-equilibrium systems has been a challenge in the history of thermodynamics and up until Onsager published his work [12] in the early days of (1931) it has not even been theoretically treated in any sort of way. This was soon to be used as a foundation for the emerging area of out-of-equilibrium thermodynamics and was set to be one of the first major contributions to it. Onsager explained certain flow forces by utilizing reciprocal relationships in non-equilibrium systems. However he weaved a special condition of local equilibrium into his theory, which meant that the derived ratios could only be used in a system near the actual equilibrium and not

driven arbitrarily far away from it. In addition to Onsager Green and Kubo separately followed up in the works [13, 14] which then became to be known as the infamously called Green-Kubo equation. This relation delivers an understanding of how to use an integral formalism over linked time-correlation functions to describe transport coefficients. But again, these mathematical descriptions lack the ability to go to a parameter regime where the system of interest is far away from being at equilibrium with the surrounding environment.

The next chapter in out-of-equilibrium thermodynamics was opened in (1951) when the Fluctuation-Dissipation Theorem (FDT) was proven by Callen and Welton [15]. Initially Nyquist found the FDT, put it down in (1928) and thereby added a versatile tool to the toolbox of statistical mechanics, which can predict the response of a system to external perturbations [16]. One major downside of the FDT in terms of the general applicability was that the system of interest still had to fulfil the condition of detailed balance. Any kind of system meets the condition of detailed balance if one cannot tell by observing the system if its temporal evolution runs forward or backward [9]. Still regardless of the reduced generality due to the restrictions of the detailed balance condition, the FDT is able to verify the predicted admittance or impedance of physical quantities, which is basically the response function of this property, by observing the intrinsic thermal fluctuations of the exact identical physical variable [16].

(1993) was the year in which an important accomplishment for the field of out-of-equilibrium thermodynamics was made, namely the introduction of the fluctuation theorem. Up until the publication [17] by Evans et al. there has not been any universal tool analysing the behaviour of a single particle and relating thermodynamic quantities on an individual particle trajectory level arbitrarily far from equilibrium. More impressively a rigorous mathematical proof of the fluctuation theorem was found one year after the release of its initial report [17]. The paper "Equilibrium microstates which generate second law violating steady states" by Evans and Searles ushered in a new era for the statistical physics community as they got a completely new, more deeply linked foundation in the form of the fluctuation theorem [18].

In fact many new equations, which paved the way for the development of stochastic thermodynamics and single particle thermodynamics, as mentioned in the title of this thesis, were deduced from the original fluctuation theorem [18]. For instance the Jarzynski equality [19], the Williams-Searles-Evans (WSE) equality [20] and the generalized Jarzynski equality [21, 22], which are going to be discussed later in more detail in section 2, can all be related to the (1993) fluctuation theorem.

The fluctuation theorem is a versatile formula stemming from statistical mechanics and describes a relationship of two time-dependent probabilities p, which correlate to the value of the thermodynamic extensive variable

$\overline{\Sigma}_t$.

$$\frac{p(\overline{\Sigma}_t = A)}{p(\overline{\Sigma}_t = -A)} = e^{At} \tag{1.4}$$

Relation (1.4) mathematically describes that the ratio of the probability $\overline{\Sigma}_t$ taking on a positive value A over the probability that the extensive quantity $\overline{\Sigma}_t$ acquires the negative counterpart $-A$ scales exponentially with the product At. This also holds true when $\overline{\Sigma}_t$ is measured during an out-of-equilibrium evolution of the system of interest. In essence the fluctuation theorem states that the bigger the system, which means the bigger the overall A would be as it is an extensive quantity, and the slower you do your evolution, translating to having a larger t, it becomes exponentially more likely that you observe a positive change of your variable $\overline{\Sigma}_t$. Just to recall, an extensive quantity is a property which scales with the size of the system, i.e., Volume V, mass m, entropy S, heat capacity C_p and free energy F to name a few.

Looking at the fluctuation theorem (1.4) one could quickly arrive at the thought that maybe this theorem suggests a violation of the second law of thermodynamics. For instance, the thermodynamic entropy change dS is an extensive variable that scales with the size of the system. So equation (1.4) would suggest that there is a parameter regime (small t or small A) where the second law, equation (1.2), breaks down. Hence, one has to be careful with the wording and reconsider the assumption on which the second law was based on. The topic lead to discussions when the very first experimental investigation of the fluctuation theorem was published by Wang et al. in (2002) with the given title "Experimental Demonstration of Violations of the Second Law of Thermodynamics for Small Systems and Short Time Scales" [23]. We already see from the title that the authors made the controversial statement that the fluctuation theorem is violating the second law. The second law of thermodynamics still holds on average. It is certainly fair to say that the 2^{nd} law has originally been stated for situations that are far from a microscopic single particle description.

We can actually attain the second law of thermodynamics by starting out with the general fluctuation theorem when we would express the mean change of a thermodynamic system of macroscopic size [24, 25]. This can be seen analogous to doing an ensemble average over the predictions given by the general fluctuation theorem. Hence, the general theorem, which is based on the different fluctuations a system can experience, can also be called a second-law-like theorem, which was the used term in the second experimental investigation of the fluctuation theorem [26]. For further reading on how the fluctuation theorem trumps the second law, I would like to advertise the first few pages of chapter 2 of book [27].

One of the many follow-up theorems which were more tailored with a formalism that was actually easy to relate to an experimental system was published

by Williams, Searles and Evans, hence the so-called WSE equality. The publication of "Nonequilibrium Free-Energy Relations for Thermal Changes" in (2008) marked the first time a theorem treated thermal fluctuations of a system [20]. With this publication [20] there was an influx in publications which not only looked at mechanical manipulations of the system of interest, which was pretty well treated at this point by the introduction of the Jarzynski equality in (1997) [19], but also at the impact of thermal changes and a generalized Jarzynski equality emerged. This generalized Jarzynski equality accounts for processes in which the system of interest can be driven far away from equilibrium by using a mechanical and/or a thermal driving agent. Both, the WSE equality and the generalized Jarzynski equality [21, 22] will be explained in a more detailed manner in section 2 of this thesis.
For more references covering other theories and more experimental guidance and a wider overview to the field of stochastic thermodynamics I would like to refer the interested reader to the wonderful review [28] by Seifert.

1.2. Motivation

The quasi-static limit is one of the fundamental limitation of classical thermodynamics. Quasi-static in this case means that the change one is performing on his ensemble of particles happens on a very long time-scale, i.e., very slowly, such that the measured thermodynamic quantities are always computed from a system which remains in equilibrium with the surrounding environment during the manipulation. This time-scale gets defined by the coupling rate of the system of interest to the environmental bath.
This assumption breaks down when we are starting to observe thermodynamic quantities of very small systems. We are able to split the problems occurring when manipulating the thermodynamics of tiny structures into two general domains. The huge impact of environmental fluctuation marks the first set of challenges that pop up. One reason for this is that the fluctuations coming from the system's surrounding are now much bigger compared to the inertias of tinier and tinier systems. Due to these compared to the small system size rather big fluctuations which are taking place all the time we are also much more prone to look at an evolution of the system which occurs very far away from the actual equilibrium condition. Being more easily in an out-of-equilibrium condition already is the second category of challenges we are experiencing.
We must not just use the classical thermodynamic tools, which would be looking at the mean behaviour and thereby leave out viable and available information of the process to properly give a description of systems of such miniature scales. In fact taking higher order moments of the entire mathematical probability distribution into account is the only way to end up with an appropriate tool to describe such a stochastic system. Not only including more moments but rather the system's full probability distribution would

mark the ideal case but it turns out that this is a very complicated procedure for the majority of systems. Assessing trajectories of a miniature set-up in which it explores state transitions far away from actually being at equilibrium with the environment is of paramount importance. But already observing such out-of-equilibrium points due to the fluctuation of the system is highly troublesome for a set of existing theories, which is also encompassing the theory of linear response. Linear response only holds for a near equilibrium condition of the system and does not imply that the system has to be in total equilibrium. The only approach which deals with the full probability distribution is the fluctuation theorem, as seen in relation (1.4). Therefore the fluctuation theorem also explains what happens if one would do a weak perturbation scheme on the set-up, even though as mentioned earlier this can lead to a significant effect on the level-scale of being in equilibrium or far away from it.

Furthermore, processes occurring in an out-of-equilibrium condition in reference to the environment are actually quite common in nature. A lot of molecule folding phenomena happen at a level, where the criteria of having an equilibrated system is not met. Getting to know the Helmholtz free energy difference for these folding processes adds a crucial understanding of how living organisms work as the myriad of biological cells in them are encompassing molecular machines, which require a non-equilibrium description [29]. Challenges are also faced when we want to investigate strong deformations and flows of solutions of dilute polymer, which can be solved by applying the fluctuation theorem out of the statistical physics toolbox [30]. Therefore knowing how thermodynamic potentials like the free energy for instance change can also be very beneficial for material science. Stretching and compression processes of metal-organic framework are often very computationally intensive to simulate using the standard molecular dynamics approach. As it turns out, the fluctuation theorem can also be of great use in computational physics to determine the breathing modes of such structures by enhancing the sampling methods in these simulations. This leads to a better understanding of the contributions due to anharmonicities and entropy in the free energy profile of these highly-flexible frameworks [31].

Moreover fluctuation theorems [19] enabled us access to the free energy, which is an equilibrium property, for mechanically driven systems and have been applied to single-molecule experiments [32], RNA-folding [33], DNA-folding landscape reconstructions [34], protein-ligand interactions in drug design [35] and electric circuits design [36]. Recently it has also been shown that fluctuation theorems play an important role in the form of quantum fluctuation relations in quantum systems as well [37].

Processes far away from the thermodynamic equilibrium can also be easily realized in the lab without going to the previously mentioned specialized set of problems. Any thermodynamic experiment which is looking at a change of a set of parameters can be turned into an out-of-equilibrium thermodynamics

experiment. In order to do that we just have to be able to change the time-scale on which we perform the parameter change to a smaller one, i.e., doing the full change quicker. At a certain speed you will witness your system being in an non-equilibrium condition and you would have to use the fluctuation theorem to still be able to give a proper description about what happens during the evolution.
The evolving field of levitated nanoparticles is opening a route to the experimental study of stochastic thermodynamics in a new parameter regime and with enhanced control abilities. Recent advances in silicon particle trapping technologies and manipulating sub-micron particles [38–40] which in particular include a flexible and fast control of the potential experienced by the particle and the temperature of its centre-of-mass motion allow us to realise and investigate thermodynamic protocols in the underdamped regime (see e.g. [41–43]). Additionally we can treat the three degrees of freedom of our trapped particle as decoupled and just look at the one dimensional case, where approximating the restoring force with a linear force is justified [44]. The levitated glass beat resembles a high quality factor system due to the decreasing collisions with the residual gas when going to vacuum. Therefore we are able to create a far from equilibrium processes more easily as the damping rate, i.e., the equilibration with the environment, happens on a longer time scale compared to overdamped systems.
This optically trapped particle acts as a one dimensional toy-model for the field of out-of-equilibrium thermodynamics as it enables us to drive the system far from equilibrium via fast temperature changes. In contrast to purely mechanical control, this yet untested situation requires a description via the Williams-Searles-Evans (WSE) fluctuation theorem [20].
In this thesis we present experimental evidence for the validity of the WSE equality. We implement thermodynamic protocols based solely on environmental temperature control of a nanoparticle. To this end we use an experimental set-up utilizing a standing wave trap in a hollow-core photonic crystal fibre (HCPCF), which is based upon the works of Grass et al. in publication [38] and will be further described in section 3.1. The temperature of the particle's effective heat bath is controlled via optical feedback cooling, please see section 3.2 for details.
Mechanical and thermal driving are the central thermodynamic processes in heat engines. Our experiment therefore complements the investigation of mechanical driving, completing the analysis of those processes from the perspective of fluctuation theorems. Moreover in this thesis we are going to show the validity of the generalized Jarzynski fluctuation theorem [21, 22] which requires simultaneous mechanical and thermal driving.

1.3. Organization of this thesis

First we will start off with outlining the theoretical background in section 2. This includes defining what kind of free energy we are interested in and how it can be expressed via the partition function as well as discussing the established equilibrium techniques of extracting thermodynamic potentials from a given system. We will be conclude this section by introducing the two kinds of fluctuation theorems which are going to be studied using the experimental setup later, namely the Williams-Searles-Evans equality and the generalised Jarzynski equality.
The thesis will then explain how we experimentally realise the thermodynamic protocols for each of the different fluctuation theorems by briefly sketching the general setup in section 3.1 which will be accompanied by detailed explanations. The explanations will cover how we realise a temperature change while keeping the mechanical part of the system untouched and how we do a simultaneous thermal and mechanical change by just tuning one parameter of the experiment. Additionally we will cover the different characterisation methods and how the measured time traces enter the computation of the final free energy differences. To round off the experimental section, we will tackle two technical questions arising from the simultaneous thermal and mechanical change we have to do in order to test the generalised Jarzynski equality in the separate subsections 3.3.1 and 3.3.2.
With everything being introduced we are already embarking into the results section 4. Therefore the results section will be split again into two main section. One dedicated to the thermal change and the other one dedicated to the simultaneous thermal and mechanical change.
The last section of the thesis will be the conclusion and will offer a summarised view of what was achieved in this work as well as include the scope of the work's contribution from a more applied perspective.
As for the appendix we will mention how we calculated the free energy difference using the partition function, how to recover the linear response theory from the WSE equality and how we converted to unitless quantities in order to evaluate them with high numerical precision on the computer. In addition we will highlight the software library that was used for the evaluation of the raw data and the thermodynamic potentials as well as discuss how to make use parallel programming for a swift and data-driven evaluation.

2. Theoretical Background

The following subsections are meant to introduce the reader to the necessary theoretical background, understanding how the different evaluation methods of the thermodynamic free energy determination came about, and why they are interesting to look at separately. Furthermore the chapter acts as a detailed explanation about the two different equalities, which will be put later to the experimental test. In addition focus will be given to the discussion of the different processes which are covered respectively by the two individual fluctuation theorems. Keeping these different use cases of fluctuation theorems in mind will be of utmost importance for the later understanding of the experimental realisations and the results in section 3 and 4.

2.1. Equilibrium and Linear Response techniques

We first look at a system surrounded by a heat bath which has a constant temperature in time. By changing an external parameter of the system, we invoke a change in the free energy of the system. Here we are speaking of the Helmholtz free energy [10, 45] which is defined as follows:

$$F = U - TS \tag{2.1}$$

In equation (2.1) F stand for the free energy of the system of interest, U represents the total inner energy stored inside the system and its change dU is governed by the heat δQ and work exchange δW as shown in the first law relation (1.1) and finally temperature T times entropy S represents the non-extractable energy of the system. Sometimes people, especially when talking about efficiencies of processes, also refer to the last term TS of relationship (2.1) as unusable energy of a system.
If we go back to the example above, we quickly see that if we manipulate an external parameter λ of the system which for instance changes the system's potential $V(\lambda)$, we establish a change in the internal energy U of the system of interest and therefore we also see a free energy difference ΔF, which points to the fact that during this process work is being extracted from the setup. This extractable work W_{mech} can then be written down as depicted in relation (2.2).

$$W_{\text{mech}} = \Delta F = F_2 - F_1 \qquad \text{with} \qquad W_{\text{mech}} = \int_{\lambda_1}^{\lambda_2} \frac{\partial V}{\partial \lambda} d\lambda \tag{2.2}$$

In equation (2.2) F_2 stands for the free energy configuration our system finished in after the external manipulation of the system's parameter λ and F_1 presents the free energy level the setup has before we begin making any changes to it. V stands for the potential energy. One way to invoke such a change from F_1 to F_2 is to decrease the finite volume of a system or increase

an external field's intensity acting on the setup. To summarize any system interaction which can be switched on and off in a controlled way can be a general parameter change of our system and invoke a free energy difference. However, equation (2.2) only holds true if the external parameter change is done infinitely slow, so fulfilling the quasi-static condition. That way the evolution of the system along some path Γ a time-dependent change of the external parameters always remains very close to equilibrium with the surrounding environment.
Thus doing the exact same controlled alteration of the system's external parameters in a finite time will change the equality in relation (2.2) to:

$$W_{\text{mech}} \leq \Delta F \tag{2.3}$$

So it can happen when we execute the process in a finite amount of time that we are not extracting all of the work, as the free energy represents the maximum amount of work a system is able to do.
Still even with keeping equation (2.3) in mind the what we will call equilibrium technique was established which uses the fact that the system of interest averages to the correct equilibrium free energy difference ΔF over many iterations of the same parameters change given the condition that the pace of the alteration is much below the coupling rate to the surrounding environment. We now make a transition from a macroscopic to microscopic perspective. So the fluctuations introduced by a process finished in a finite time acting on the system are assumed to be uniformly distributed and are expected to average away over many repeats. The validity of these assumptions for our experimental system will be tested later. Comparing the actual free energy difference calculated by partition function theory (appendix subsection A.1) to the equilibrium method, as seen in equation (2.4), used at different execution speeds of the same state change, will later show the limitation of this technique.

$$\Delta F = \langle W_{\text{mech}} \rangle \tag{2.4}$$

The brackets represent the ensemble average, in this case of the thermodynamic extracted work W. As stated earlier, always being in equilibrium is a key requirement in order to apply the formula (2.4) correctly. In the middle of the 20^{th} century during the emergence of the statistical physics community, scientists became increasingly interested in looking at the linear response of the system of interest. Especially processes which involved a thermodynamic protocol executed in the near-equilibrium regime led to ground-breaking discoveries and no longer restricted the system to be absolutely in equilibrium all the time [46]. Thus, the system of interest was also allowed to explore states which were in close vicinity to the equilibrium condition. The linear response theory formulation for calculating the free energy difference by measuring the work of the system in an near-equilibrium

condition is equation (2.5) [47].

$$\Delta F = \langle W_{\text{mech}} \rangle - \frac{\beta}{2} \left\langle W_{\text{mech}}^2 - \langle W_{\text{mech}} \rangle^2 \right\rangle \tag{2.5}$$

After the publication of the Jarzynski equality [19] it turned out that linear response theory in this case is the first order Taylor expansion of the fluctuation theorem relating work extraction of a mechanically driven system with the free energy difference. However, equation (2.5) can be independently derived from linear response theory to the best of our knowledge [47]. The main difference between the linear response theory and the full Jarzynski fluctuation theorem, which can be seen in relation (2.6), is that the Jarzynski equality is also valid for processes which reside even far-from-equilibrium, while the linear response theory will fail at this point due to the near-equilibrium restriction.

$$\left\langle e^{-\beta W_{\text{mech}}} \right\rangle = e^{-\beta \Delta F} \tag{2.6}$$

The Jarzynski equality only accounts for mechanical changes of the system and relates non-equilibrium mechanical work production with a free energy difference. Additionally the inverse heat bath temperature β remains constant during the entire change of the system.
The linear response formula, like in relation (2.5), incorporates the first two moments of the probability distribution, namely the mean and the variance of the work. However higher moments of the probability distribution are important to evaluate the extracted work correctly. We will especially see a striking difference between free energy difference of the linear response theory and the fluctuation theorem approach, which takes into account the entire probability distribution, i.e., all higher moments, when we do a manipulation of our system far away from equilibrium.
Furthermore deriving a linear response formalism, as seen in equation (2.5), can also be done for the Williams-Searles-Evans (WSE) equality [20] and the generalized Jarzynski equality [21, 22]. In the appendix subsection A.2 we will see how to recover the linear response for a thermally driven system starting out with the WSE-equality, which will be introduced in more detail in the following subsection 2.2.

2.2. William-Searles-Evans equality

Going back to the Jarzynski equality shown in equation (2.6), we see that the inverse temperature β of the thermodynamic system enters the relation as a constant. Thereby it needs to be stressed that the Jarzynski equality only allows for mechanical work to kick your system into an out-of-equilibrium condition.
However with the new control abilities in levitated optomechanics, which will be shown in section 3, and the precise and if desired fast manipulation of the

centre of mass (COM) temperatures of laser trapped nano-particles, we need to come up with a description which can account for thermal changes acting as a driving agent in our experimental system of interest. To visualize the described process of realising a free energy difference ΔF by a nano-sized system coupled to a heat bath which is changing its temperature, I refer the reader to figure 1.

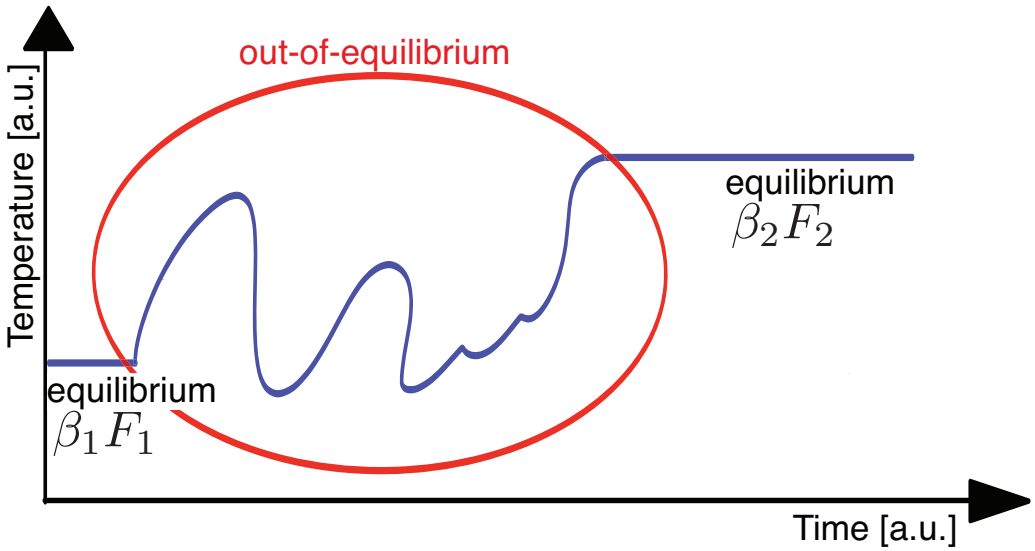

Figure 1: Illustrated thermal change: the blue line represents the change in temperature and the red ellipse highlights the arbitrary change, where the system has no time to equilibrate, if the change is done in a finite time. Everything that is not included in the red ellipse is a state in equilibrium with the environment.

The thermal fluctuations in the environment can occur at an arbitrary the time scale. Therefore we are looking for a concept which provides the proper characterization of thermodynamic potential changes, like the free energy difference, no matter how far the system is driven away from equilibrium by thermal fluctuations or an external thermal driving agent in a controllable manner.
Luckily there exists an ready-to-apply formalism which was deduced from the original fluctuation theorem [17] and explains any kind of arbitrary thermal driving scheme on arbitrarily fast time scales. This new fluctuation theorem is called the Williams-Searles-Evans (WSE) equality [20]. The derivation of the WSE equality was initially published by Williams et al., hence the name WSE equality, under the title "Nonequilibrium Free-Energy Relations for Thermal Changes". The authors focused to derive a simple formalism which is ready to be applied to experiments and their final expression can be seen

in equation (2.7).

$$\beta_2 F_2 - \beta_1 F_1 = -\ln\left(\left\langle e^{-\int_0^\tau \dot{\beta}(t) H_0(\Gamma(t))dt}\right\rangle\right) \tag{2.7}$$

On the left side of equation (2.7) β_1 and β_2 represent the starting and the finishing temperatures of the connected heat bath respectively. Similarly F_1 and F_2 are the free energy at the beginning and at the end of the thermal change of the environment coupled to our system. So we can say that on the left side of relation (2.7) we see a sort of normalised free energy difference. In particular in this expression of the state difference the free energies are normalised to the corresponding inverse temperatures of the heat bath. On the right hand side of relation (2.7) we see that we integrate from the beginning of the change $t_{\text{start}} = 0$ to the end of the manipulation protocol $t_{\text{end}} = \tau$. Inside the integral we have the time-derivative of the environmental inverse temperature $\beta(t)$, which marks the hallmark of the WSE fluctuation theorem as we are no longer bound to a constant heat bath and can realise processes with a thermal driving agent. This change of the inverse temperature $\dot{\beta(t)}$ is then multiplied with the initial Hamiltonian H_0, which is provided with the phase space coordinate $\Gamma(t)$, i.e., the position and velocity information of our system of interest. So after integrating over a single realization of such a thermal change protocol, we apply the exponential function to the final value of one integrated trajectory. By repeating this procedure we get multiple values for the same process and thereby built up our thermodynamic ensemble even though we are just looking at one particle. This method of constructing a so-called time-like ensemble is justified as long as the ergodic condition is fulfilled. After we have our time-like ensemble we can take the average and thereby a non-linear averaging effect as we have an exponential weighting in place. Taking the natural logarithm of the averaged value leaves us with the normalised thermodynamic potential difference, i.e., the free energy difference.
For our purposes the Hamiltonian H_0 consists of the standard kinetic term, which we take into account since we operate in an underdamped system, plus the potential term $V(x)$ as can be seen in equation (2.8).

$$H_0 = \frac{p^2}{2m} + V(x) \tag{2.8}$$

In relation (2.8) p is the momentum with $p = m \cdot v$, m is the mass of the in our case trapped nanoparticle, ω is the fundamental frequency of the harmonic oscillator and x represents the positional information of our system.
Analogously to the Jarzynski equality formulation, equation (2.6), we can define a thermal work θ and rewrite the WSE equality, relation (2.7), in the

following form:

$$\beta_2 F_2 - \beta_1 F_1 = -\ln\left(\left\langle e^{-\theta}\right\rangle\right) \qquad \text{with} \qquad \theta = \int_0^\tau \dot{\beta}(t) H_0(\Gamma(t)) dt \tag{2.9}$$

Hence we have the following formulation for the equilibrium technique in the situation of thermal driving:

$$\beta_2 F_2 - \beta_1 F_1 = \langle\theta\rangle \tag{2.10}$$

and the following alteration for the linear response (appendix section A.2):

$$\beta_2 F_2 - \beta_1 F_1 = \langle\theta\rangle - \frac{\beta}{2}\left\langle \theta^2 - \langle\theta\rangle^2 \right\rangle \tag{2.11}$$

Equations (2.10) and (2.11) will be used for the evaluation in the results chapter 4.1.

2.3. Generalized Jarzynski equality

Looking back onto the introduced theoretical frameworks we see that with the WSE equality [20], as seen in relation (2.7), and the Jarzynski equality [19], as seen in equation (2.6), we can only cover special cases of external driving agents acting upon our system of interest. The Jarzynski equality accounts for mechanical fluctuations in our system and the WSE equality treats manipulations of the thermal environment of the apparatus. But if we want to describe a process which involves a mechanical and a thermal control of the experiment, we simply fail to do so with the current introduced theoretical concepts.
However many processes happening in nature and in our daily lives would require this special treatment, involving a thermal and a mechanical change. A basic heat engine cycle, like the Otto cycle, is comprised of compressing a gas by confining its volume via the movement of a piston, which would mark the mechanical manipulation part of the system, and a rapid burning of the ignited mixture of fuel and air, which would represent the thermal parameter change in the engine.
Thus having a fluctuation theorem in place which can treat a simultaneous change of a mechanical and a thermal parameter of the system, would allow us to give a proper description for the evolution of the free energy F during a general thermodynamic process like a heat engine cycle.
In order to create a better understanding of what processes we would like to describe with the introduced formalism, I would like to point to figure 2 visualizing such a proposed process.

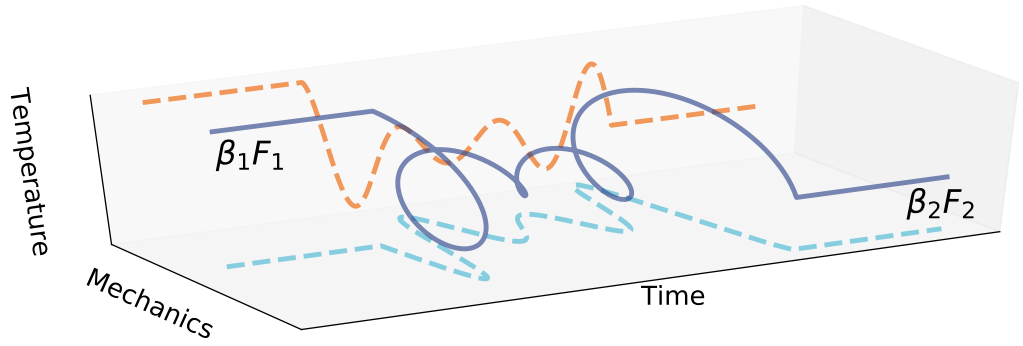

Figure 2: Illustrated simultaneous thermal and mechanical change: the blue line represents the simultaneous change happening in temperature of the heat bath and the potential energy of the system, i.e., the system's mechanics. In between the straight lines at the beginning and the end lies the arbitrary change, where the system has no time to equilibrate, if the change is done in a finite time. The orange line is a projection of the blue one and represents temperature against time. The projected turquoise line shows mechanical change against time.

In the following equation (2.12) we see a new kind of fluctuation theorem which was first published by Chelli et al. [21, 22]. This very fluctuation theorem is also referred to as the generalized Jarzynski equality, as it encompasses a treatment for not only mechanical but in addition thermal fluctuations of the system of interest.

$$\beta_2 F_2 - \beta_1 F_1 = -\ln \left\langle e^{-\int_0^\tau (\dot{\beta} H + \beta \dot{H}) dt} \right\rangle \tag{2.12}$$

In relation (2.12) we still calculate a normalised free energy difference on the left hand side of the equality in relationship (2.12). Again each of the free energies of the beginning F_1 ($t_{\text{start}} = 0$) and end state F_2 ($t_{\text{end}} = \tau$) are normalised with respect to the inverse heat bath temperature connected to the system. On the right hand side of the equation we encounter the same non-linear averaging as we did for the WSE equality and the original Jarzynski equality. Thereby we again have to have a system which meets the needs of ergodicity in order to use the generalized Jarzynski equality.
Coming back to the differentiating key component of the generalized Jarzynski equality in equation (2.12) we see when we look closely at the term after the integral that we do not only integrate over the time derivative of the time-dependent inverse heat bath temperature but we also include the time derivative of the Hamiltonian, which is not constant for the entire protocol any more but rather undergoing an evolution in time on its own via a change in the mechanics of the system. This means that the effective potential term contributing to our Hamiltonian is changing over time. Therefore our

Hamiltonian which describes our system susceptible to a mechanical external driving agent can be seen in relationship (2.13).

$$H = \frac{p^2}{2m} + V(\lambda, x) \tag{2.13}$$

In relation (2.13) we see that the external time-dependent mechanical drive in the form of $\lambda(t)$ is directly affecting the potential term of our Hamiltonian. For instances for an optically trapped silica particle, which can be approximated as an underdamped harmonic oscillator, we can introduce such a manipulation by effectively changing the spring constant k, which is directly followed by a modification in the fundamental frequency of the system since we have $\omega = \sqrt{\frac{k}{m}}$. We will later see in the experimental realisation subsection 3.3 that the spring constant change translates to a laser power change of the beam trapping the nanoparticle.
If we now connect the Hamiltonian of our system in equation (2.13) with the generalized Jarzynski equality in relationship (2.12), we see that the integrand is in the fluctuation theorem is basically the total derivative of the inverse heat bath temperature times the Hamiltonian, i.e., βH. This full derivative can be seen in relation (2.14).

$$d[\beta H] = \underbrace{\beta \cdot \left(\frac{\partial H}{\partial x} dx + \frac{\partial H}{\partial \dot{x}} d\dot{x} \right)}_{\text{heat}} + \underbrace{\beta \cdot \frac{\partial H}{\partial \lambda} d\lambda}_{\text{mech. work}} + \underbrace{\frac{\partial [\beta H]}{\partial \beta} d\beta}_{\text{thermal control}} \tag{2.14}$$

Heat is the energy change due to random fluctuations, which the system experiences due to the surrounding environment. Mechanical work is the change of inner energy due to the manipulation of an external system parameter λ. Thermal control is applied by a change of the environmental temperature, i.e., β as control parameter.
Additionally we can relate the generalized Jarzynski equality as seen in equation (2.12) to the original Jarzynski equality as seen in equation (2.6). By taking the special case of the generalized non-equilibrium work relation and applying a pure mechanical change to the system we effectively end up with the integral over the time-derivative of the Hamiltonian undergoing a change in the effective potential due to a change of the external parameter $\lambda(t)$ and this equals the calculated mechanical work if we look back onto equation (2.14). Additionally one can relate it to the definition of mechanical work in equation (2.2), since a manipulation of an external parameter λ is only affecting the potential term V of the Hamiltonian, i.e.:

$$\left\langle e^{-\int_0^\tau \beta \dot{H} dt} \right\rangle = \left\langle e^{-\beta \int_{\lambda_1}^{\lambda_2} \frac{\partial V}{\partial \lambda} d\lambda} \right\rangle = \left\langle e^{-\beta W_{\text{mech}}} \right\rangle = e^{-\beta \Delta F} \tag{2.15}$$

For the generalized Jarzynski fluctuation theorem, relation (2.12), one can again define a generalized work Λ, analogously to the Jarzynski equality formulation, equation (2.6), and rewrite the generalized Jarzynski equality in the following form:

$$\beta_2 F_2 - \beta_1 F_1 = -\ln\left(\left\langle e^{-\Lambda}\right\rangle\right) \qquad \text{with} \qquad \Lambda = \int_0^\tau (\dot{\beta} H + \beta \dot{H}) dt \tag{2.16}$$

Hence we have the following formulation for the equilibrium technique in the case of a mechanical and thermal drive:

$$\beta_2 F_2 - \beta_1 F_1 = \langle \Lambda \rangle \tag{2.17}$$

and the following alteration for the linear response:

$$\beta_2 F_2 - \beta_1 F_1 = \langle \Lambda \rangle - \frac{\beta}{2} \left\langle \Lambda^2 - \langle \Lambda \rangle^2 \right\rangle \tag{2.18}$$

Equations (2.17) and (2.18) will be used for the evaluation in the results chapter 4.2.

3. Experimental Realisation

This chapter will introduce the reader to the world of experimental levitated optomechanics in an abbreviated manner. The experimental setup was developed in the Master thesis [48] and Doctoral thesis [49] of D. Grass and published in [38]. We will introduce the necessary background with a focus on the requirements and modifications for the presented work.
By first establishing an understanding for how we are able to optically trap a nanometre-sized particle by using a laser beam we will lay the foundation for the subsequently following sub-chapters. Thereupon we will respectively deal with the realisation of exposing the trapped nanoparticle to an external thermal driving agent and having simultaneous external mechanical and thermal manipulation of the experimental system.

3.1. Experimental Setup

A simplistic version of the optical trapping setup can be seen in figure 3.

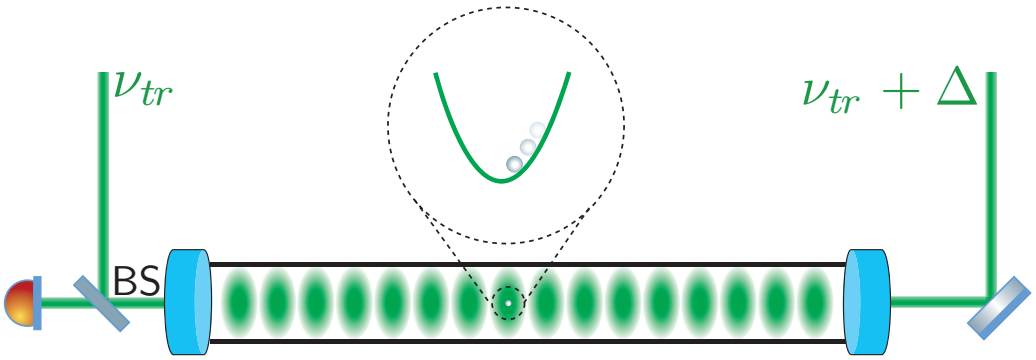

Figure 3: Levitation of nanoparticles: Two equally polarized laser beams with a wavelength of $\lambda_{\mathrm{tr}} = 1064\mathrm{nm}$ are aligned into the hollow core photonic crystal fibre (HCPCF), which create a standing wave trap inside a HCPCF. This provides a harmonic trap for the nanoparticle, as seen in the inset. Detuning one of the two beams in frequency, i.e., $\Delta \neq 0$, leads to the movement of the optical trap along the fibre. Hence the original purpose of the setup was to offer a novel nanoparticle loading mechanism from room pressure for ultra high vacuum experiments, i.e., an optical nano-silica conveyor belt. Part of the scattered light by the particle exits the system via a 90/10 beamsplitter (BS) and is guided onto a photo diode for a balanced detection scheme.

At the centre of the experimental setup lies the hollow core photonic crystal fibre (HCPCF). Two vacuum chambers are connected via the $\sim$ 20cm long fibre, which performs like an optical conveyor belt transporting particles of submicron size in-between the chambers. Theses individual chambers can

be separately controlled in pressure. However, for the purpose of the experiments in this Master thesis, we will always have an intermediate connecting valve between these two chambers opened to guarantee equal pressures. The levitation of the particles is realised by taking a laser beam from a Nd:YAG laser system with a wavelength $\lambda_{\mathrm{tr}} = 1064\mathrm{nm}$, splitting the beam via a beam splitter, aligning each of the respectively created arms through an acoustic-optical modulator (AOM) and finally position them such that they are entering the HCPCF each on one side of it. Thereby a standing wave trap is created inside the fibre. Trapping of nano-particles is established in one of the vacuum chamber when both laser beams have a frequency ν_{tr}. By having both arms passing through respective AOMs we have the ability to control the detuning of one beam in relation to the other. Hence, we can establish an optical conveyor belt. When the detuning is turned on, one beam has ν_{tr} and the other one has $\nu_{\mathrm{tr}} + \Delta$ with $\Delta \neq 0$ and we move the entire standing light wave trap including the particle which is already and still trapped in the standing wave inside the HCPCF.
The standing light wave presents itself to the dielectric nanoparticle as an array of trapping potential wells and one such well can be described by the following Rayleigh formalism [50] in equation (3.1).

$$U = -\alpha \frac{I(r,z)}{2\varepsilon_0 c} \qquad \text{where} \qquad \alpha = 4\pi\varepsilon_0 a^3 \frac{\varepsilon - 1}{\varepsilon + 2} \tag{3.1}$$

α in relation (3.1) represents the polarizability of the particle, $I(r,z)$ is the intensity distribution with $r = \sqrt{x^2 + y^2}$ and z standing for the displacement in the axial direction inside the HCPCF, ϵ_0 denotes the vacuum permeability, ϵ corresponds to the dielectric constant, c is the speed of light and a is the radius of the particle, which additionally has to meet the following condition $a << \lambda_{\mathrm{tr}}$. At this point if one is interested in the entire detailed calculation including the expression of the total intensity distribution, I refer to page 8ff. of the previous Master thesis [48] by David Grass about the trapping setup. By nebulizing an isopropyl alcohol solution, which is containing the nano-silica, into one of the chambers at room pressure the initial source of flying particles is established. Due to the high environmental damping and the gradient force F, which the particle sees due to the array of potential wells in the standing light wave and can be seen in relation (3.2), the silica particle is locked in the adjacent perimeter of the intensity maximum and oscillates around it.

$$\vec{F}_{grad} = -\vec{\nabla} U \tag{3.2}$$

For small oscillations, as typically occur, the harmonic oscillator approximation can be employed to describe the motion of the trapped particle using the standard Langevin dynamics [51] and the equation of motion is written down for the one dimensional case in equation (3.3).

$$m\ddot{x} + m\gamma_{\mathrm{p}}\dot{x} + m\omega^2 x = F_{\mathrm{therm}} \tag{3.3}$$

m represents the mass of the harmonic oscillator in relationship (3.3) and γ_{p} is directly related to the pressure inside the setup and stands for the rate of damping experienced by the particle. This damping force due to the collision of the gas particles with the nano-sized particle is sometimes also called Stokes force [52]. ω stands for the fundamental mechanical frequency of the oscillator and F_{therm} is the Brownian force noise [53, 54], which is governed by the temperature of the surrounding thermal environment. In this instance, the standard room temperature $T_0 = 293\mathrm{K}$ enters the force noise description.

3.2. Thermal Change

To realise a thermal change in the proposed experimental system, we utilize a velocity feedback based cooling system, which basically consist of an intensity-modulated second laser exerting radiation pressure on the trapped nanoparticle. This so-called feedback laser allows us to optically simulate an effective heat bath for the centre of mass degree of freedom of the system and can be schematically seen in figure 4. The temperature of the effective heat bath can be controlled via a gain setting inside the feedback system. In order to create the thermal protocols, which tuned the particle's effective centre of mass temperature, we had to implement a time-dependent gain into the feedback system for this thesis, which we will explain in detail later.

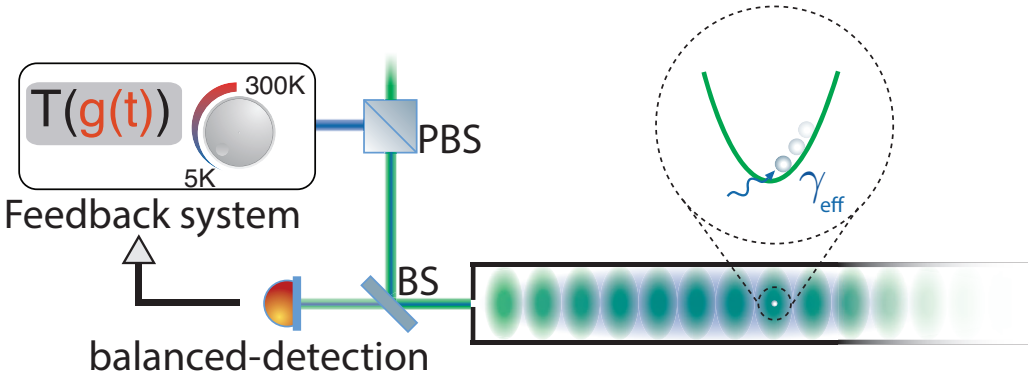

Figure 4: Velocity feedback cooling of nanoparticles: The position read-out provides the input for the feedback system. The by $\frac{\pi}{2\omega_0}$ in respect to the particle's frequency ω_0 delayed and amplified signal controls the centre-of-mass motion of the trapped particle via the radiation pressure of a second laser (blue) and thereby creates an environment with an effective damping term γ_{eff}. Adjusting the gain $g(t)$ directly manipulates the intensity of the feedback and thereby the temperature T of the centre-of-mass degree of freedom.

The feedback system is based on a laser amplifier, which outputs a constant power at a wavelength of $\lambda_{\mathrm{fb}} = 1064\mathrm{nm}$. Since the velocity feedback laser is

operating at the same wavelength as our trapping laser, we take several precautions to minimize the possible crosstalk of the two light beams. First we put the feedback laser to a completely different frequency ν_{fb}, when it passes through an AOM for the intensity modulation. In fact the trapping beams ν_{tr} and $\nu_{\mathrm{tr}}+\Delta$ lie at a very different frequency, such that no interactions with the feedback system at ν_{fb} occur at relevant time scales. Experimentally we realise that by taking a different order of the AOM for the feedback light and the trapping light. Secondly, we are keeping interference between the trapping and the feedback beam to an absolute minimum by always having an orthogonal polarization between the two lights, hence the polarizing beam splitter (PBS) for coupling in the feedback light into the HCPCF in figure 4. The velocity feedback scheme makes use of the scattering force experienced by the trapped when you shine light onto it. We take the positional readout signal of our underdamped harmonic oscillator at a pressure of around 1mbar and feed it to a delay line, which is set to delay the signal by a $\frac{\pi}{2\omega_0}$ shift in respect to the fundamental frequency ω_0 of our oscillating submicron silica. The feedback laser light is then modulated in intensity using the $\frac{\pi}{2\omega_0}$ delayed position signal, which is the velocity information since we have a high-Q oscillator [49]. Thus whenever the particle tries to go up the slope of the optical trap towards the direction of the feedback laser, we increase the intensity of the feedback light, hence the scattering force which the particle sees increases as well and thereby decrease the kinetic energy of the particle and when the nanoparticle climbs up the potential well in the opposite direction we decrease the feedback light in order to attain a lower scattering force push experienced by the confined oscillator. This implies that when the particle resides at the centre of the trap, we have a average displacement by the feedback light of half of the maximum applicable scattering force in the system. So we effectively introduced a new feedback force, i.e.:

$$F_{\mathrm{fb}} = m\gamma_{\mathrm{fb}}\dot{x} \tag{3.4}$$

With this feedback system in place we have to modify our equation of motion in order to cope with the new evolution of our system's dynamics. Every time we apply the velocity feedback scheme we effectively reduce the width of the distribution of the center of mass motion of our submicron sized oscillator by introducing an additional damping term γ_{fb} via the modulation of the scattering force in the experiment. The additional feedback damping term γ_{fb} manifests itself in front of the velocity term of the Langevin equation of our system. Thus we basically now have an effective damping factor γ_{eff} created by the addition of the environmental damping γ_{p} due to the ongoing collisions of our tiny silica with the surrounding residual gas particles and the newly introduced feedback damping mechanism γ_{fb}, i.e., $\gamma_{\mathrm{eff}} = \gamma_{\mathrm{p}} + \gamma_{\mathrm{fb}}$. γ_{fb} is comprised of a gain parameter $g(t)$, which accounts for the time-dependent amplification happening in the FPGA and a delay time τ, which

enters the overall phase shift ϕ applied the acquired position signal. For all the experiments presented in this subchapter τ was chosen to be $\tau = \frac{\pi}{2\omega}$. Additionally γ_{fb} is inversely proportional to the oscillation frequency ω of the particle, i.e.

$$\gamma_{\text{fb}} = g(t)\frac{\sin(\phi)}{\omega} \qquad \text{with} \qquad \phi = \omega\tau = \frac{\pi}{2} \tag{3.5}$$

For the derivation of the additional damping in presence of feedback, I refer to next section 3.3, as we will perform a general analysis of the feedback control, i.e., $\phi \neq \frac{\pi}{2}$.
The newly developed Langevin equation [51] describing our harmonic oscillator with a velocity dependent feedback scheme is then:

$$m\ddot{x} + m(\gamma_{\text{p}} + \gamma_{fb})\dot{x} + m\omega^2 x = F_{\text{therm}} \tag{3.6}$$

Let us now take a closer look at the thermal force noise term F_{therm} of equation (3.6):

$$F_{\text{therm}} = \sqrt{2\gamma_0 m k_B T_0} \cdot \xi_t \tag{3.7}$$

ξ_t is representing the underlying noise behaviour, i.e., white noise. Equation (3.7) shows that even though we are changing the effective temperature of the centre of mass motion of the particle, we do not scale the introduced force noise with it, assuming the feedback does not introduce noise. This is a signature of the cold damping effect, which is analogous to the direct velocity feedback. As we change the effective damping γ_{eff}, we also change the effective temperature T_{eff} accordingly, i.e., $T_{\text{eff}} \cdot \gamma_{\text{eff}} = \text{const.} = T_0 \cdot \gamma_0$ with $\gamma_0 = \gamma_{\text{p}}$.
Based on these considerations we can calculate the effective temperature T_{eff} with feedback as:

$$T_{\text{eff}} = T_0 \cdot \frac{\gamma_{\text{p}}}{\gamma_{\text{p}} + \gamma_{\text{fb}}} \tag{3.8}$$

For an alternate derivation of equation (3.8) please see appendix section A.3. In relationship (3.8) T_0 represents the environmental temperature of the residual gas inside the HCPCF, which is $T_0 = 293\text{K}$ in our case.

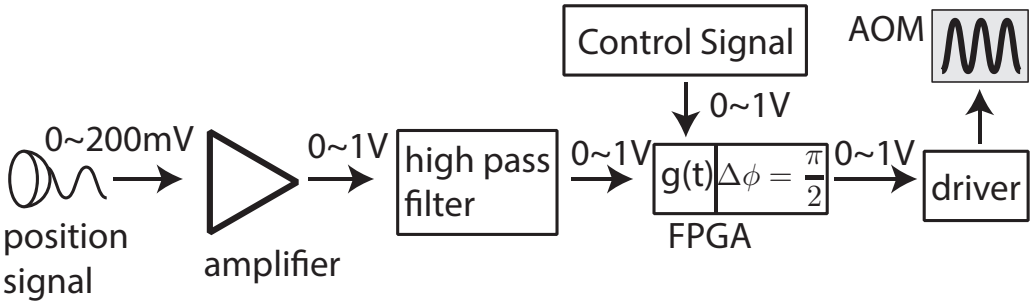

Figure 5: Schematics of the feedback system: The voltage produced by the photo diode is taken and amplified by 10dB. The signal is then filter by a high pass filtered and fed into the FPGA. The FPGA has an additional input for feeding in a control signal to modulate the gain $g(t)$ for the feedback. The delay setting on the FPGA is equalling to a relative phase shift $\phi = \pi/2$. The total delay considering non-reducible electronic delay due to the technical implementation of the feedback is $\tau = 5\pi/(2\omega_0)$ This delayed signal is then passed on to the AOM driver.

The time evolution of the gain was controlled by a function generator which was set to output a pulse signal with a modified leading and trailing edge. So as can be seen in figure 6, the system was allowed to equilibrate for several milliseconds to the initial room temperature state and then the cooling was ramped up to a certain degree in a defined leading edge time frame and kept at this point for an additional several milliseconds to allow the particle to equilibrate to the cooled end state, which was again followed by a ramp down to the room temperature state with a set trailing edge time, which was the same as the leading edge time. After the overall 25ms long protocol was finished, it started directly again.
The thermal change control protocol depicted in figure 6 was designed to have a protocol time of $\tau = 2.26$ms, which was the slowest thermal and ensured that we had an equilibrium reference for the faster thermal changes done later. In order to built up the necessary statistics for the later evaluation of the free energy difference by the WSE equality as seen in equation (2.7), we repeated the experiment 15000 times, while keeping the time during which the thermal change is happening always constant at $\tau = 2.26$ms. So the repeated signal on the function was convenient to record a single long time trace and later digitally cut the long trace into the appropriate chunks given by the 25ms periodicity condition leaving just the thermal ramp for evaluation. The actual realised temperature change, which was realised over the course of different time scales for the thermal protocols ranging from $\tau = 2.26$ms to $\tau = 22.6\mu$s (appendix section B), can be seen in dependence of the feedback gain in figure 7. In figure 7 the cooling factor of 8.3 was obtained by taking the ratio of the variance of an equilibrated room temperature state (g=0) over an equilibrated cooled state (g=1), i.e., $\sigma^2_{\text{room}}/\sigma^2_{\text{cooled}}$.

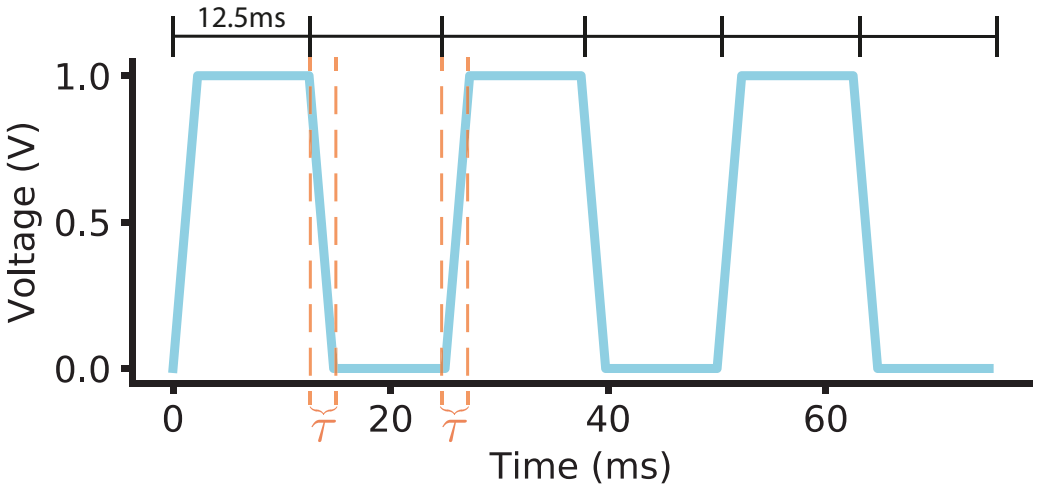

Figure 6: Thermal driving protocol: The turquoise line is the feedback gain controlling signal for the slowest thermal ramp implemented in the experiment, i.e., $\tau = 2.26$ms. This signal is send to a FPGA, which controls the gain of the delayed signal. When the signal is 1 maximal cooling is applied to the trapped particle and when it is 0 the system is at room temperature.

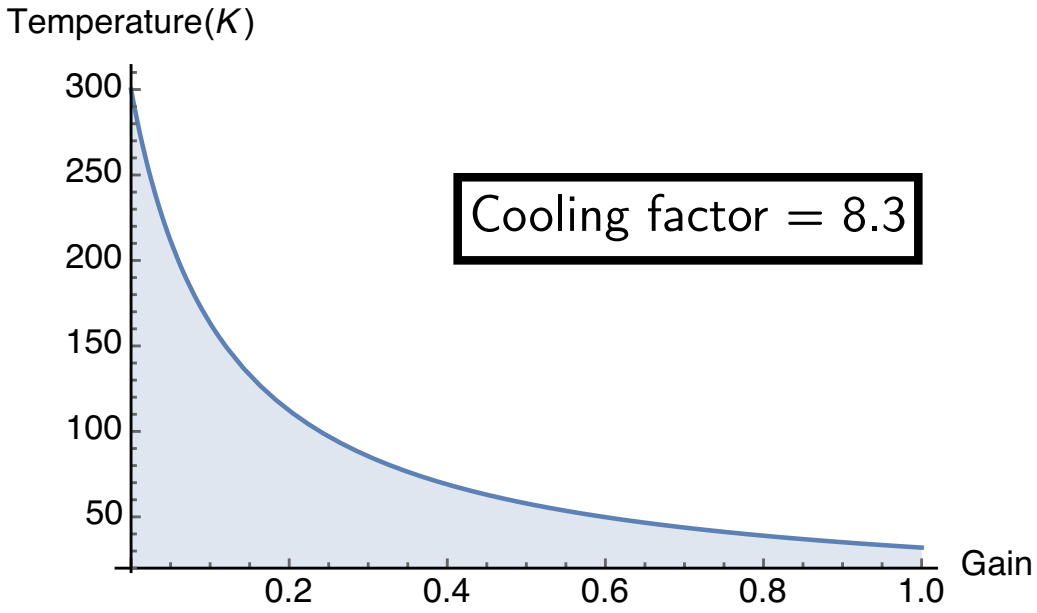

Figure 7: Model of the temperature evolution: Centre-of-mass temperature of the trapped particle is plotted against the gain g. In the actual experiment the gain is ramped up by a function generator from 0 to 1. At $g = 1$ the feedback system has a cooling factor of 8.3, which translates to $T = 36.14$K.

This measurement scheme was done for different protocol times. Starting off

with the slowest one of $\tau = 2.26$ms, where the introduced thermal change happens at a level where the particle is always in equilibrium with the optically engineered heat bath and going all the way to the shortest protocol of $\tau = 22.6\mu$s, which is a hundred times faster and places the particle into a non-equilibrium environment. Raw time traces, where the equilibrium data in between the runs is already removed up to a residual minimum, can be seen in figure 8.

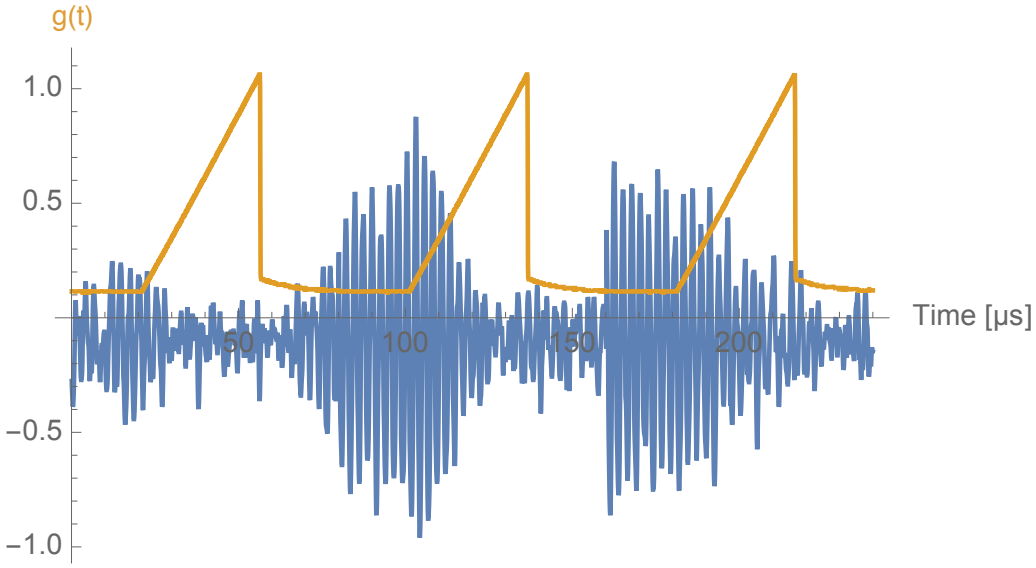

Figure 8: Raw data after removing equilibration time after thermal ramp: blue is proportional to the yet uncalibrated particle position and orange the gain control signal. The protocol time for the thermal change is $\tau = 36\mu$s. We see a significant reduction of the centre of mass motion of the nanoparticle as soon as we start ramping up the gain of the feedback system. After the ramp was executed and we go back to zero gain again, we see that the particle starts to equilibrate to the room temperature environment due to collisions with the residual gas molecules.

Before we go on and feed the data to the fluctuation theorem as seen in equation (2.7), we manipulate the data in post-processing by filtering the trace with a digitally generated bandpass filter with a bandwidth of 200kHz, compared to a particle linewidth of (6383 ± 163)Hz, around the fundamental frequency of the trapped nanoparticle $\omega_0 = (305164 \pm 7787)$Hz and cut out the time trace, where the system is undergoing a thermal change, i.e., where the gain is not constant in time. The filter window was purposely chosen that broad to be sure to keep the chance of introducing any kind of artefacts due to filtering to an absolute minimum and to be confident that we are not dragging low and high frequency noise into the evaluation of the final results. Figure 9 shows some of the measured trajectories after that processing and calibrating the system to nanometres:

To get from a voltage χ on the photo diode, where χ is the recorded signal for the particle position in volt, to nanometres x_{nm} we use thermal equilibrium, because only then are we able to harness the power of the equipartition theorem [9], equation (A.8), and calculate a proportionality constant c with $x_{\mathrm{nm}} = \sqrt{c}\chi$. We take a long time trace when feedback is turned off. We then calculate the power spectral density (PSD) of the data and fit it with the fitting equation (3.13). From the fit we extract the area, which corresponds to the variance of the motion inside the potential well, and the fundamental frequency ω. Additionally we assume the mass for the sphere given by a (supplier-specified) material density of $\rho = 1.950\mathrm{g/cm}^3$ and diameter of $d = 969\mathrm{nm}$. The thermal equilibrium is estimated to be at $T_0 = 293\mathrm{K}$. Finally we feed all the parameters to equation (3.9), to obtain the conversion factor $c = 398.28324\mathrm{nm}^2/\mathrm{V}^2$ for position calibration.

$$c = \frac{k_B T_0}{m\omega^2 \langle x^2 \rangle} \tag{3.9}$$

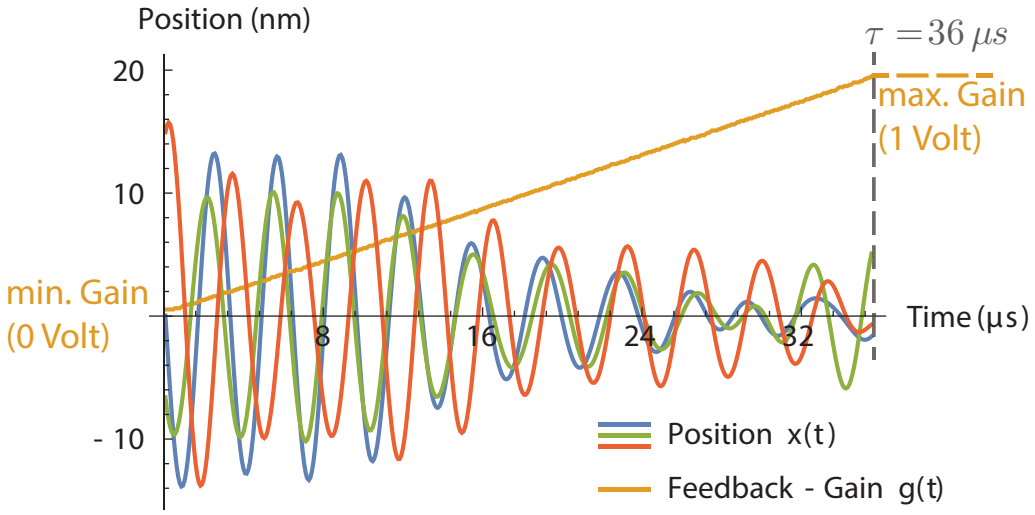

Figure 9: Filtered and cut time traces: Red, green and blue lines represent the bandpass filtered position data of the trapped particle, while undergoing a thermal change controlled by gain signal in orange, which is ramping up from 0 to 1 in a protocol time of $\tau = 36\mu$s. A total of 15000 traces was recorded to create a sufficiently large ensemble to evaluate the different free energy difference calculation methods later.

We then take each trace, calculate the velocity of the trapped nanoparticle by doing a one-point derivative and feed the constructed phase space coordinate $\Gamma(t)$ into the Hamiltonian, as in equation (2.8). We numerically integrate over the Hamiltonian as well as the temperature change like specified in the WSE equality, seen in relation (2.7). The outcome of this evaluation is discussed in chapter 4.

3.3. Thermal and Mechanical Change

In the following we are going to explain how we realise a simultaneous change in the system's potential and effective environmental temperature by just tuning one knob in our experimental setup by exploiting the feedback delay to particle frequency relation. This requires a modification to our previously presented scenario. The modified version of the experimental framework can be seen in figure 10. This setup allows to implement a power change which leads to a change in the system's potential and particle oscillation frequency ω_{eff} over a wide range. The change in frequency results in a modified effect of the delayed feedback which leads in turn to a change in the effective environmental temperature T_{eff}. The relation between these quantities is shown below in equation (3.12).

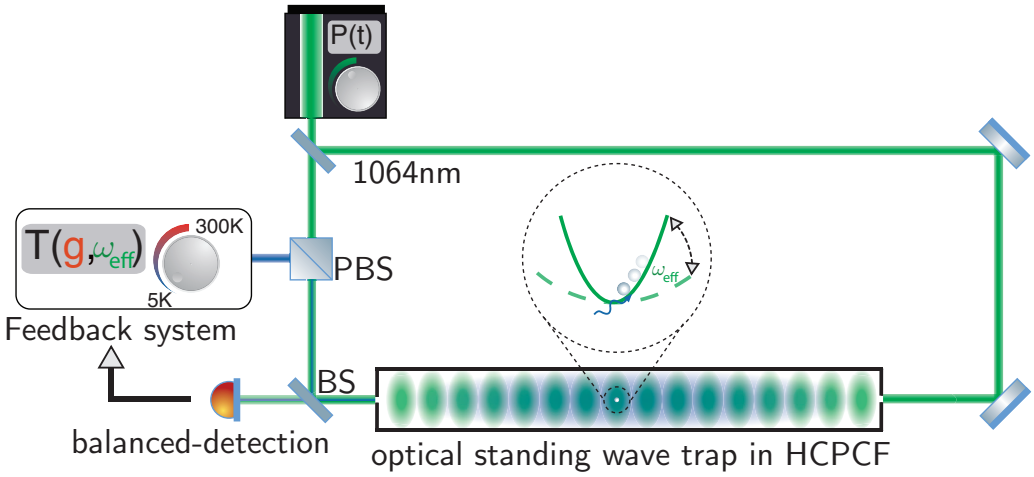

Figure 10: Levitation, cooling and mechanical manipulation of nanoparticles: A laser with a wavelength of $\lambda_{\text{tr}} = 1064nm$ (green) is split into two equally polarized beams, which create a standing wave trap inside a HCPCF. This provides a harmonic trap for the nanoparticle, as seen in the inset. Part of the scattered light is guided onto a diode for balanced detection. The detection signal provides the input for the velocity feedback system. It controls the centre-of-mass motion of the trapped particle via the radiation pressure of a second laser with a wavelength of $\lambda_{\text{fb}} = 1064nm$ (blue). The overall power $P(t)$ of the trapping laser (green) can be controlled. Since the feedback system is dependent of the power related fundamental frequency ω of the oscillator and the gain g is constant, we have a simultaneous mechanical and thermal change by controlling the trapping power $P(t)$.

We see in figure 10 that we now added a control capability, namely we can arbitrarily change the laser power of the trapping beam. A change in the laser power will have several implications for the dynamics of the system:
We see that we now have an effective fundamental frequency ω_{eff} in the equa-

tion of motion (3.11), rather than a constant ω_0 for the harmonic oscillator. This is due to the fact the trapping power P and fundamental frequency ω of the trapped nanoparticle fulfil a direct relationship, namely $\omega_{\text{eff}} \propto \sqrt{P}$. Additionally we can make out that the feedback system is not only gain dependent but also dependent on the effective frequency ω_{eff} of the confined submicron sized particle.
Let us analyse how the feedback system is altering the equations of motion when we operate it during a power change. For that let us start off by considering a force which is governed by a time-delayed position $x_{t-\tau}$ and can be controlled by a gain g. Since we are experimentally operating in the underdamped regime, i.e., $Q >> 1$, we are allowed to use the high-Q approximation and further assume that we can ignore the damping and the force noise over a few oscillations. This then leads to a treatment of our levitated particle as a sine function $x(t) = x_0 \sin(\omega t)$. By using the following trigonometric relationship $\sin(x-y) = \sin(x)\cos(y) - \cos(x)\sin(y)$, the force the feedback exerts is:

$$F_{\text{fb}} = g \cdot x_{t-\tau} \approx g \cdot x_0 \sin(\omega_{\text{eff}}(t-\tau)) = x_t \underbrace{g \cos(\tau\omega_{\text{eff}})}_{\omega_{\text{fb}}^2} - \dot{x}_t \underbrace{g \frac{\sin(\tau\omega_{\text{eff}})}{\omega_{\text{eff}}}}_{\gamma_{\text{fb}}} \tag{3.10}$$

We now introduce a fixed time-delay of $\tau = \pi/(2\omega_0)$. Thus, our equation of motion to describe the movement of the nanoparticle gets a modified damping term γ_{fb} and an additional optical spring ω_{fb} and looks like the following:

$$\ddot{x} + \left(\gamma_{\text{p}} + \frac{g}{\omega_{\text{eff}}} \sin\left(\frac{\pi}{2}\frac{\omega_{\text{eff}}}{\omega_0}\right)\right)\dot{x} + \left({\omega_{\text{eff}}}^2 - \omega_{\text{fb}}^2\right) x = F_{\text{therm}}/m \tag{3.11}$$

In equation (3.11) we significantly changed the term responsible for the damping occurring in our system. Let us focus on the sine term, because the gain g of the feedback system will be set to a constant value for all the experiment doing a simultaneous mechanical and thermal change. With the sine term in equation (3.11) we have an accurate expression of how our feedback system depends on the fundamental frequency of the oscillating particle. For an ideal velocity feedback cooling scheme, we would like to delay our signal exactly by a $\frac{\pi}{2\omega_{\text{eff}}}$ shift, which we used in the previous section on thermal driving. However to convert this $\frac{\pi}{2\omega_{\text{eff}}}$ shift into a time which can be programmed into a delay line, the introduced phase $\tau\omega_{\text{eff}}$ is then dependent on the frequency ω_{eff} of the optomechanical oscillator.
The experimental protocol is based on a power modulation of our trapping laser light. This first of all changes the optical trap stiffness of the particle's potential and thereby detunes the fundamental frequency of the trapped silica, which equals a change in the system's susceptibility. More importantly

we have feedback cooling for the entire time and the detuned frequency also detunes our feedback system as we set the delay to be perfectly delaying the position signal by a $\frac{\pi}{2\omega_0}$ shift at $t = 0$, so before we do any power changes on the trapping beam. Thus as we change the trapping power and as a result of it the particle frequency gets altered and we start to do worse and worse cooling of our confined oscillator, thereby introducing an effective thermal change to our system. Figure 11 shows a schematic of the power modulation and its effect on the feedback system.

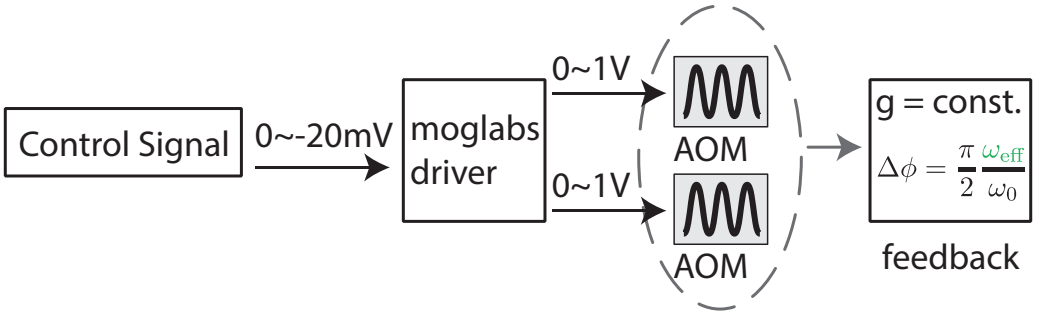

Figure 11: Schematics of the power control and its influence on the feedback: The function generator sends a control signal with a ramp between 0mV and −20mV to the Moglabs XRF421 driver, which in turn passes it on the AOMs modulating the trapping power. This creates an effective frequency of the particle ω_{eff} and since the delay is tuned to a constant value optimized for $\omega_{\text{eff}} = \omega_0 = \omega_{\text{end}}$, we do vary the amount of feedback cooling applied to the system while we manipulate the power.

The signal controlling the power of the the trapping laser beam was again a modified pulse signal produced by a function generator in such a way that the leading and trailing edge time of our pulse signal could be modified in order to achieve different cooling times.
Overall one entire cycle, in which the trapping power was linearly ramped up, kept up for several milliseconds, then turned off again, had a period of 25ms and we actually use the same signal as seen in figure 6. The several milliseconds static time periods in which the power $P(t)$ was not altered were implemented in order to allow the particle to equilibrate to the surrounding environment and therefore start every change in temperature from a thermal state. However the effective temperature change was not done in a linear fashion as the effective temperature T_{eff}, to which the particle tries to equilibrate, is governed by equation (3.12). For a derivation of the effective centre of mass temperature of the particle please see appendix section A.3.

$$T_{\text{eff}} = T_0 \cdot \frac{1}{1 + g(t)\sin\left(\frac{\pi}{2}\frac{\omega_{\text{eff}}}{\omega_0}\right)} \tag{3.12}$$

The ramping time of the leading and trailing edges of the pulse signal was adapted for every protocol ranging from 22.6μs, as the fastest cycle, to 2.26ms,

which was the slowest rate at which the trapping power was altered (appendix section B). By starting from a lower power setting and ramping up the trapping power we achieved a change of the trapping frequency ω_{eff} by 11%. Consequently relation (3.8) with $\omega_0 = \omega_{\text{end}}$ shows that we start off with less effective feedback cooling because we are not delaying the velocity exactly by $\pi/(2\omega_{\text{eff}})$ but by $\pi/(2\omega_{\text{end}})$. As we ramp up the power we start to do more optimal feedback cooling until we arrive at the optimal delay again when $\omega_{\text{eff}} = \omega_{\text{end}}$. So by just changing the power of the optical trap we simultaneously invoke a mechanical and thermal change of the particle.

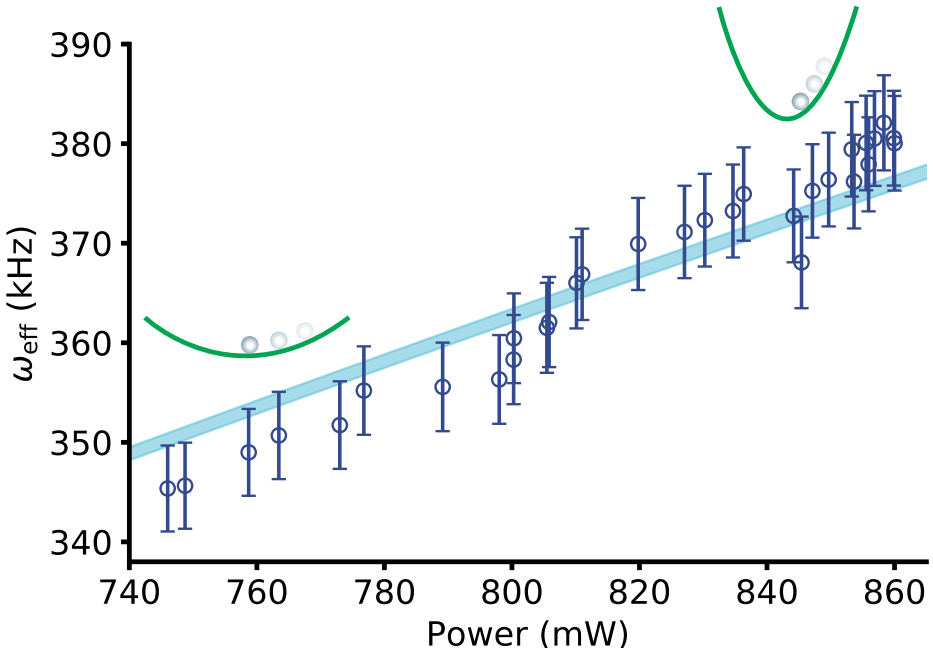

Figure 12: Power-dependence of the fundamental frequency with feedback turned off: Frequency of the trapped particle is plotted against the optical trapping power in blue. The error bars for the blue data points were devised through a Gaussian error propagation of the known experimental short time scale fluctuation on the laser power (1.25%) and the PSD fitting error. In turquoise we see the fit which utilizes the square root dependence of laser power to fundamental oscillator frequency. The inserted schematics of the harmonic trap show an exaggerated depiction of the harmonic trap when doing the power change.

In order to confirm our ideas how the system's dynamics evolve when we do a change of the trapping laser power, we first reduced the power in steps in the same power range, where we later do a continuous ramp. At each step we allowed the particle to equilibrate to the new environment for several seconds and then recorded a time trace of 1 second. In figure 12 we see the measured relationship between the trapping power and the levitated

oscillator's frequency and we can clearly make out that we are already in the linear slope of the square root relation between the two variables.
The frequencies of the confined submicron sized particle in figure 12 were determined by computing the single power spectral densities (PSD) of the separate time traces at each step along the power change and fitting the corresponding formula for an harmonic oscillator, which was taken from the supplement of publication [39] and can be seen in equation (3.13). One of these fits can be seen in figure 13.

$$f(\omega) = \frac{C}{(\omega^2 - \Omega_{\text{eff}}^2)^2 + \omega^2 \gamma_{\text{eff}}^2} \tag{3.13}$$

C in fitting equation (3.13) represents a free parameter when doing the fit and stands for $C = aT_{\text{eff}}\gamma_m$ with a being a calibration constant.

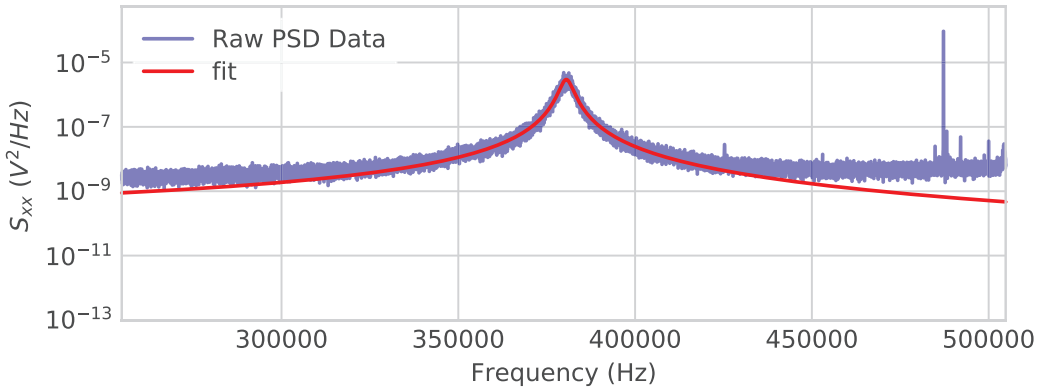

Figure 13: Example fit used for determining the frequency: violet shows the PSD calculated from the raw voltage data taken of the position detection and red shows the fit of the generated PSD. The fitting equation can be found in equation (3.13).

We also want to test our experimental model for the temperature change. To this end we are mainly interested in the the cooling factor η, which can be determined in several ways. The cooling factor is usually defined by a ratio of the cooled and uncooled state of the system, but the selection of representative variables which we can choose in order to determine a cooling factor is plentiful. The different approaches can be seen in equation (3.14).

$$\eta = \frac{\gamma_{\text{eff}}}{\gamma_{\text{p}}} = \frac{\gamma_{\text{cooled}}}{\gamma_{\text{uncooled}}} = \frac{A_{\text{uncooled}}}{A_{\text{cooled}}} = \frac{\sigma^2_{\text{uncooled}}}{\sigma^2_{\text{cooled}}} = \frac{T_0}{T_{\text{eff}}} \tag{3.14}$$

In relation (3.14) A represents the area under the mechanical peak in the PSD and σ^2 stands for the variance of our particle's distribution. The area is computed by summing up the frequency array of the PSD in the region

of 80kHz around the mechanical peak. The gammas were deduced by fitting the mechanical peak with the fitting equaiton (3.13). The variances get computed by using already filtered time traces. To investigate how much the window size of the bandpass filter plays a role when computing variances, we compared 40kHz filter window to a 90kHz one. The 40kHz sized one follows the peak when we change the power and the 90kHz filter window was choosing to be static with the center of it lying at 365kHz. Thus the mechanical peak of the frequency changing oscillator was always inside the static 90kHz window. The 90kHz sized bandpass filter was later also used to filter the actual thermodynamic protocols. The cooling factor η in equation (3.14) is also a very important quantity for the later evaluation of the generalized Jarzynski equality, because the cooling factor is directly proportional to the observed inverse temperature change, which we later have to integrate over in the fluctuation theorem, i.e., $\eta(t) \propto \beta(t)$.
To experimentally check the cooling factor evolution and compare the different evaluation methods we reduced the power in steps in the same power range, where we later do a continuous ramp. At each power step we took a time trace of 1 second with feedback cooling on and off, while leaving enough time to the particle beforehand to equilibrate to the new power and feedback setting. Figure 14 shows an illustration of the entire experimental procedure and its time line.

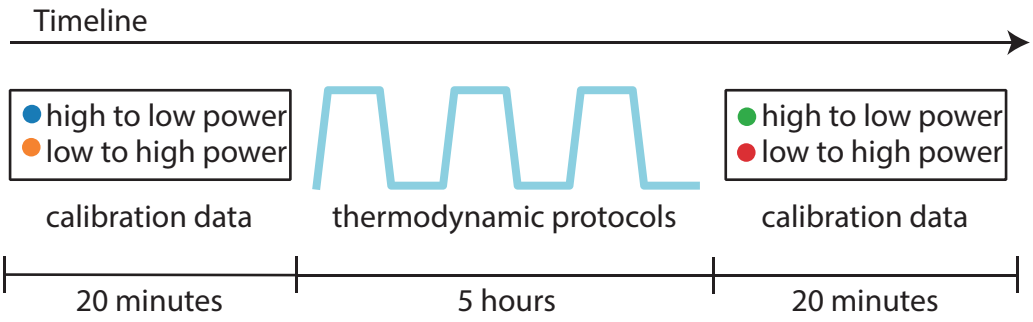

Figure 14: Temporal arrangement of the experimental protocol: First we take a calibration data set by going down in power in steps and afterwards going up to the starting value again, while we switch feedback on and off at every point. Each step we take 1 second of data, after the particle has calibrated to new feedback and power control setting. Then it takes us 5 hours to take 15000 trajectories of each of the 9 different protocol times. Finally, we take a second calibration dataset by reiterating the procedure done for the first calibration dataset.

In figure 15 we see how the different evaluation methods compare in respect to each other. Furthermore we see that we have differences between the different datasets, as the blue and the orange dataset were taken before we carried out the continuous power change ramps for each different protocol time, which took all in all 5 hours, and the green and the red datasets were

recorded after we finished all of the different ramps. More specifically the blue data was measured by going from the highest power setting to the lowest and the orange dataset was acquired while changing the power in steps from the lowest to the highest laser intensity. Similarly, the green data evolves temporally from high to low power and the final red dataset from low to high optical power.

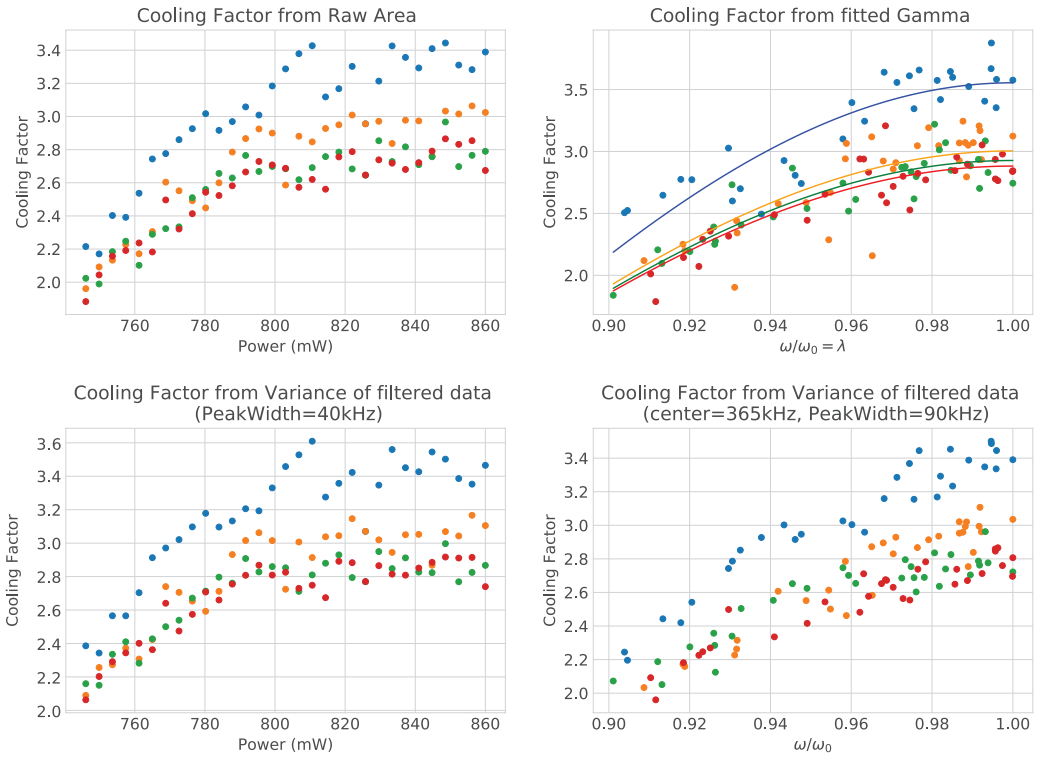

Figure 15: Comparing different methods of calculating the cooling factor evolution: For all plots: the blue and orange datasets were taken before the continuous runs and green and red after the ramps, see figure 14. In the upper left graph we see $A_{\text{uncooled}}/A_{\text{cooled}}$. A is the area under the mechanical peak in the PSD. The lower left and lower right shows $\sigma^2_{\text{uncooled}}/\sigma^2_{\text{cooled}}$. The variances σ^2 are computed from filtered time traces, i.e., a 40kHz filter window following the peak on the left and a 90kHz static filter window centred around 365kHz on the right. The upper right graph depicts $\gamma_{\text{cooled}}/\gamma_{\text{uncooled}}$. γ is determined by fitting the peak as in figure 13. Historically all four plots had power as an x-axis. We opted to extract our model of the thermal change, i.e., $\beta(t)$, via the PSD fitted gamma (upper right graph). Therefore we changed the x-axis to a relative frequency change ω/ω_0 to suit the final evaluations demands requiring a direct relation between environmental inverse temperature β and mechanical change parameter λ. For cross checking purposes we also changed the x-axis of the lower right to a relative frequency change one. The fit in the upper right graph uses equation (3.12) to express T_0/T_{eff}.

Additionally in figure 15, we see that the choice of the filter window width for the bandpass filter has a negligible contribution to the overall analysis of the system's dynamics. Even when we change from a narrow filter (40kHz bandwidth), as seen in the lower left graph of figure 15, to a broad filter setting (90kHz bandwidth), as seen in the lower right graph of figure 15, we do not see a change in the characteristic behaviour or the overall level of the calculated cooling factors. The fits of the spectrally evaluated datasets in the upper left plot of figure 15 were later used to test a model for the thermal change which takes place due to the detuning of the feedback system by the trapping power manipulation. However we can see that the fits can not really be taken with confidence since the data points which are used to construct the fit have a significant spread due to the fluctuations happening in the system.

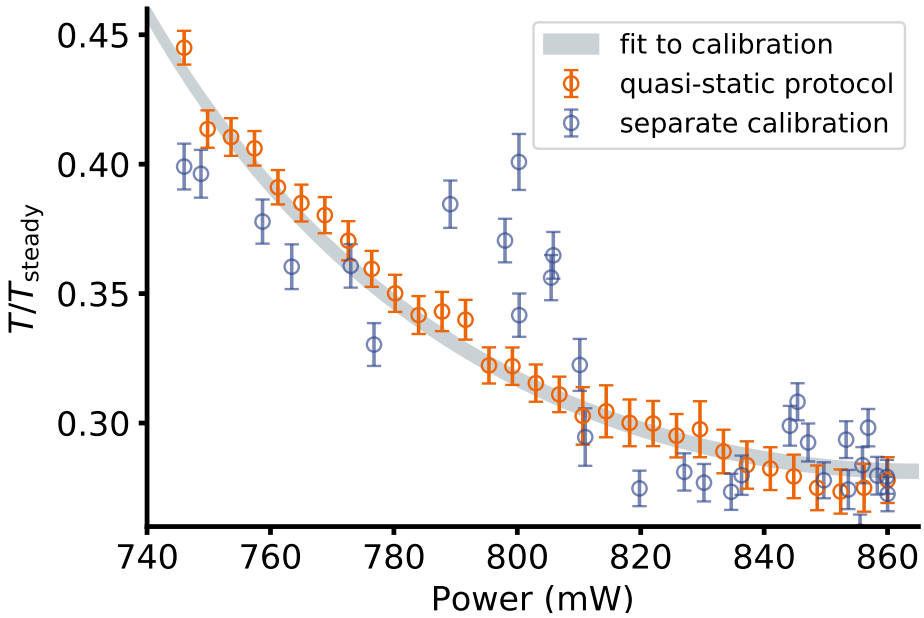

Figure 16: Power-dependence of the reciprocal temperature: Two different methods for the inverse cooling factors are plotted against the laser power in the optical trap. The blue error bars are calculated by doing Gaussian error propagation of the PSD fitting errors and the known experimental drifts, like the laser power (1.25%). For the orange data points the error bars consist of the statistical measures extracted from the ensembles that are used to compute each point and the known experimental short time scale drifts again. The fit is generated using relationship (3.12) and the blue data set is from the calibration set shown in the upper left plot of figure 15. The orange data is extracted by reconstructing an ensemble evolution along the power ramp of the slowest protocol ($\tau = 2.26ms$) and taking out time slices.

In order to tackle this problem and restoring confidence in the fitted models, we looked for an additional way to gain access to the thermal model of the system undergoing a power change. We utilize the actual continuous power ramps that are used in the quasi-static version of the actual protocol. We then reconstruct the position distributions of the system along the continuous simultaneous mechanical and thermal change by taking out the position information at a certain point in time of all the individual trajectories recorded for the longest protocol time. By this method we were able to recreate an ensemble at every chosen point in time. From the newly acquired distributions we could compute the cooling factor via the variance, equation (3.14). Hence we are able to follow the ensemble's evolution in respect to the thermal change. This method allowed for a variable time resolution when looking at the change. The only limiting factor for how many distributions we could calculate along the time line of the change was given by the actual sampling rate we used to record the time traces. The result of this undertaking can be seen in figure 16 and we are able to conclude a very good match to the fit, which was shown earlier in the upper left plot of figure 15. In order to compare figure 16 with figure 15 one must be aware of the inverse relation between the cooling factor η and the reciprocal temperature expression T/T_{steady}, i.e., $1/\eta = T/T_{\text{steady}}$.

3.3.1. Calibration of power-dependent position read-out

Since we change the overall laser power trapping the particle, we also see a reduction in the light, which is scattered and detected by the balanced photo-diode read-out scheme. Therefore we have to account for the general sensitivity change of the detection system.
By taking the monitor output channel of our balanced photodiode, which is the direct current (DC) intensity measured by the diode, and utilizing it to renormalise our gathered data, we were able to account for the power dependent detection efficiency. To this end we recorded 1 second long time traces, which were taken at specific power setting with velocity feedback on and off, as can be seen in the legend in figure 17. We divided the raw signals by their respective monitor signals and computed the PSD of this newly monitor output normalised particle signal. This is shown in figure 17. The peaks around 370kHz represent the oscillations of the particle in the harmonic trapping potential when feedback is off and on. We can clearly make out that when feedback is turned off the cold damping effect takes place and the height of the peak gets reduced. Additionally we see a striking noise peak close to 500kHz, which is electronic noise coming from the detection system. The important feature in figure 17 is that the background level stays constant in relation to the introduced power change, which shows that rescaling by the monitor output accounts for the power dependent detection efficiency change.

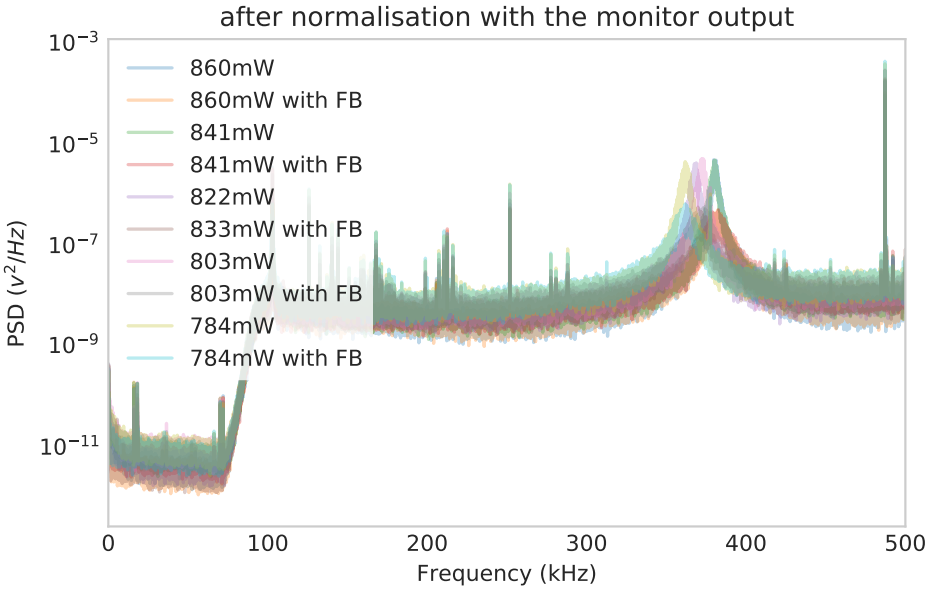

Figure 17: Rescaled PSDs of cooled and uncooled states at different power settings: Power Spectral Densities of the raw time traces at different optical trapping power setting are plotted. With FB in the legend is short for describing that the velocity feedback system was switched on at this stage. The PSD were rescaled by using the output signal of the monitor channel of the balanced photo diode.

Another cross check we performed in order to be sure that the rescaling with the monitor output covers the effect was to compute the area under PSD, so basically the variance of our particle's distribution, multiply it with the square of the particle's frequency and multiply it as well with the effective damping coefficient. This value is supposed to be constant when we change the power.

$$A\omega_{\text{eff}}^2\gamma_{\text{eff}} = \frac{2\pi k_B T_{\text{eff}}\gamma_{\text{eff}}}{m} = \text{const.} \qquad \text{with} \qquad A = 2\pi\frac{k_B T_{\text{eff}}}{m\omega_{\text{eff}}^2} \tag{3.15}$$

This quantity, as can be seen in equation (3.15), should be the same for the feedback cooled and uncooled case, since area under the PSD and the damping coefficient scale inversely in respect to each other and should therefore cancel the change in the system's dynamics. This is directly related to the fact that when we do the velocity feedback, which is analogous to cold damping, we have $\gamma \cdot T = \text{const.}$.

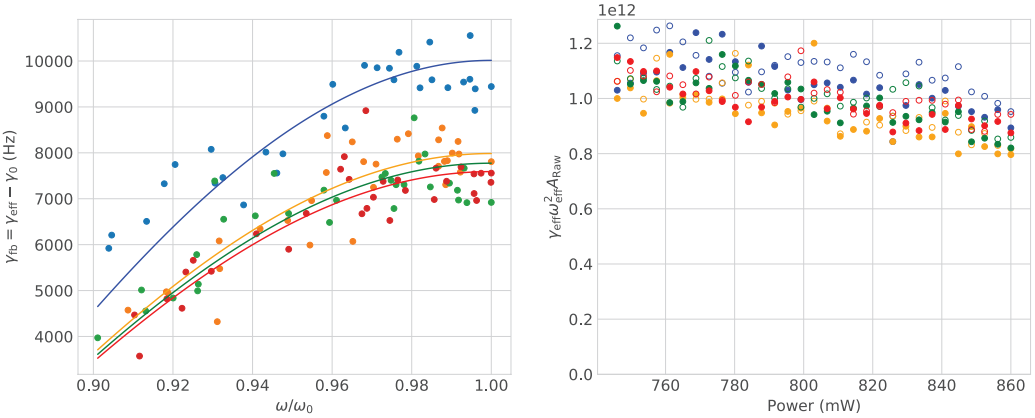

Figure 18: Cross-checking the monitor output rescaling method: For both plots, the blue and orange datasets were taken before the continuous runs and green and red after the ramps, see figure 14. The fit for the left plot was done by using the sine relationship of the effective damping γ_{eff} to the mechanical change, see denominator of equation (3.12). The drift observed in the datasets over the 5 hours long measurement time are used to estimate the range of expected outcomes (results section 4.2).

The resulting cross check is depicted on the right side of the figure 18, while the left hand side features the behaviour of the additional damping contribution γ_{fb} due to the detuned feedback system. In the left graph of figure 18 the filled dots represent uncooled data and the shallow ones stand for evaluated time traces with velocity feedback turned on, i.e., cooled data. Similar to the previous plots, blue and orange points were taken before the actual continuous power ramps and green and red data was recorded after these runs. While there are significant observable fluctuations between the individual datasets and the cooled data point might not always lie exactly on top of the uncooled data point, there is a high enough correlation between the two sets. Furthermore, we see a slight increase in the collective value of the evaluated data, which points to a drift which is not yet understood.
In the final evaluation we include the range of observed cooling factors in the theory estimates and assume that the monitor output calibration method is sufficient for the evaluation.

3.3.2. Influence of power-change on bath temperature

Since we actively modulate the trapping laser power of our system we could be prone to observing an additional degree of heating of the trapped silica particle, which has been observed in different optical trapping setups for different particle sizes and laser powers in [55]. Thereby we would have an unaccounted contribution to the thermal change that we do for testing the fluctuation theorem. Note that this cannot explain the systematics in figure 18 as the temperature would be higher at high powers.

In order to test if our experimental system suffers from such heating we reduced the power of the optical trap in steps from 860mW all the way down to almost 200mW and analysed the equilibrium temperature. We took time traces at each step after the particle equilibrated to the new power setting and extracted the distributions from these time traces.

$$\langle x^2 \rangle = \frac{k_B T}{m\omega^2} \tag{3.16}$$

By using equipartition theory [9, 10], as seen in equation (3.16), we were able to calculate a measure which is directly proportional to the centre of mass temperature of the silica beat. The measurement shows no indication of any significant additional heating. We conclude that the uncooled constant value corresponds to room temperature, which we use for the calibration of the position in figure 19.

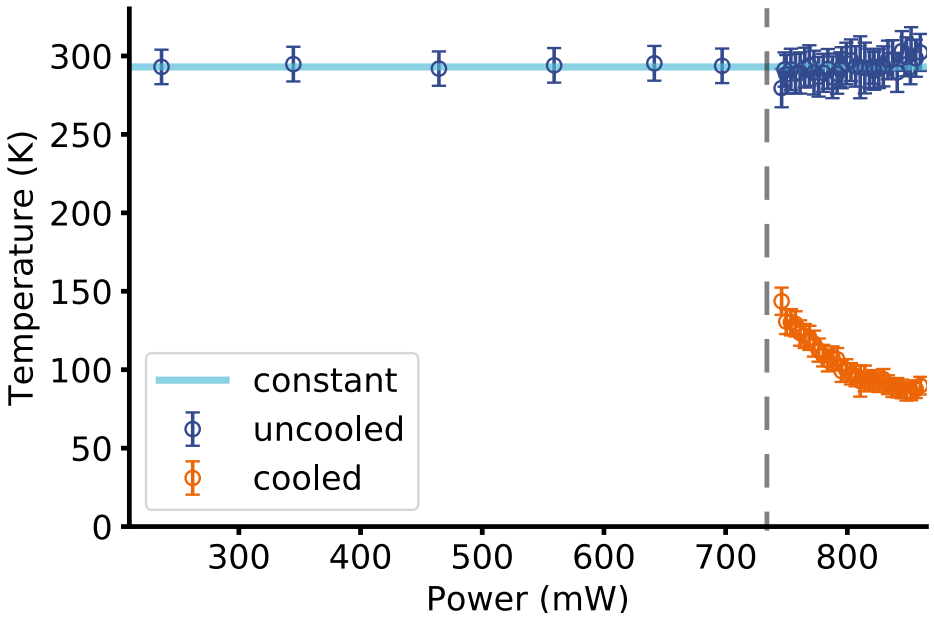

Figure 19: Power-dependent absorption heating mechanism: The orange data is the same one as the orange one showed in figure 16, i.e., feedback cooled power-dependent temperature. The blue data is consisting out of two data subsets of data. Slow power ramps ($\tau = 2.26$ms) with the feedback system turned off were used for the points above the orange data and evaluated the same way as the orange data points in figure 16. The blue data points in the region between 700mW to almost 200mW are evaluated by taking steps in power and recording time traces after the particle had a chance to equilibrate to the environment. The turquoise horizontal line is a constant function at 293K.

Conclusion

In this chapter we have seen, that we have proper models for the change in temperature and frequency during our experimental protocols, by comparison to measurement. We have characterised the drifts of the model parameters over the measurement time of 5 hours. We have provided the necessary detection calibration. In combination this now enables the evaluation of the fluctuation theorem.

4. Results

This chapter will focus on presenting the main results of this Master thesis. In order to create a clear understanding and explanation of the final plots proofing the validity of the different fluctuation theorems, I structured the section into two subsection.
In the first subsection we focus on a fast thermal protocol and compare our experimental results with the Williams-Searles-Evans (WSE) equality as shown in equation (2.7) in chapter 2.2 on page 13 when the system encounters thermal driving. I will compare the results of the WSE equality with the results from an equilibrium evaluation (2.10) and the linear response theory (2.11).
In the second subsection of this chapter we apply both thermal and mechanical driving far from equilibrium to our system. We will show the consistency of the generalized Jarzynski equality, as can be seen in relation (2.12) in chapter 2.3 on page 15 and compare to the equilibrium evaluation (2.17) and the linear response theory (2.18). While the generalised Jarzynski equality can be used for any kind of state change, we chose one that is specifically suited for our system. We demonstrate that for this specific protocol that both the thermal and the mechanical contribution of the driving are significant.

4.1. Thermal Drive by Feedback Cooling

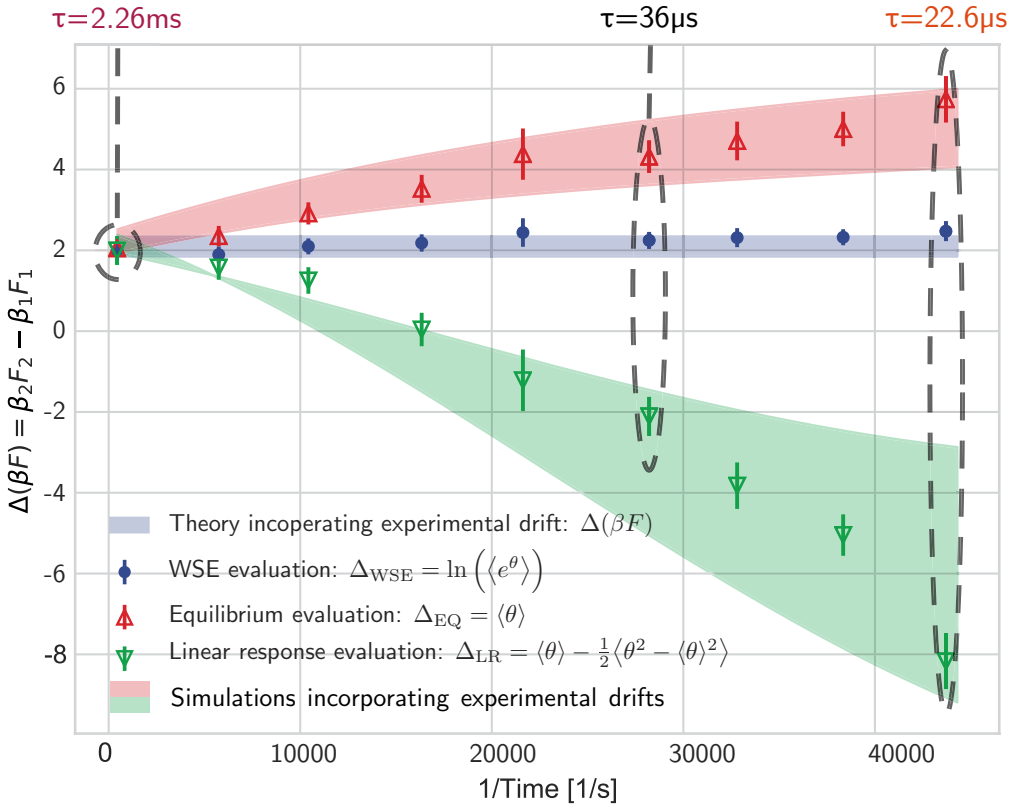

Figure 20: Comparison between experiment evaluation methods and theory for purely thermal driving: The blue data shows the actual theory predicted values compared to the blue data points which are based on the Williams-Searles-Evans evaluation. The red and the green rely on the equilibrium and the linear response method and depict a strong deviation from theory once we go to faster driving times. For a detailed discussion of the methods, please take a look at the residual text of this subchapter. The dashed grey markers show the fastest protocols, the slowest protocol and an intermediately timed protocol, which corresponds to the data taken in figure 9.

In the following we will present the results for the thermal driving experiment as presented in section 3.2. We compare the estimated difference in normalised free energy evaluated by three different methods: the Williams-Searles-Evans (WSE) equality, an evaluation assuming an equilibrium change and linear response theory.
In figure 20 we plot the normalised free energy difference against the inverse protocol time, which quantifies how far the system is driven away from equilibrium. On the hand left side the thermal change happens very slowly over 2.26ms, i.e., the system stays near equilibrium. We increase the speed for

the same thermal protocol by two orders of magnitude. This amounts to a 22.6μs execution for the fastest protocol at a damping rate of (6383 ± 163)Hz.
First we compare the data evaluated based on the WSE equality to the theoretical expectation. The blue shaded area in figure 20 represents the results obtained from the partition function calculation (appendix section A.1). This specifies the true normalised free energy difference. The range of values covered by the area was determined by taking the drift of the cooling factors over the measurement time of 5 hours into account.
The blue dots represent the evaluation of our measurement data using the Williams-Searles-Evans (WSE) equality, equation (2.7), as described in the last paragraph of subsection 3.2. For each data point we evaluated 15000 trajectories each one undergoing the same thermal change fixed with a specific protocol time τ. The error bars for this data are obtained by error propagation of the trajectory ensemble variance and the short term drift of the laser power (1.25%) typical for our experiment.
We find a very good agreement between the experiment and the theoretical prediction. We want to stress that the blue area is not a fit but a calculation using independent experimental parameters of the system. Thus the WSE equality indeed provides a suitable good method to determine free energy differences even for system thermally driven far from equilibrium.
Now we compare the evaluation based on the equilibrium assumption (red) and linear response theory (green) to the results presented before.
The red and the green area represent again predictions for the expected experimental outcome of the two methods, respectively. The range covers the bounds of the predictions considering, as before, the drift of the cooling factors. These predictions are based on simulations of the physical system, which are evaluated equivalent to the data points:
The red triangles mark the equilibrium evaluation method, which is basically a linear average over the integrated trajectories as seen in equation (2.10).
The green up-side-down triangles stand for the evaluation using the linear response theory as seen in equation (2.11) and derived starting from the WSE fluctuation theorem (appendix section A.2). These results have been obtained using the same dataset as for blue WSE evaluation. The error bars have been calculated equivalently by error propagation of the trajectory ensemble variance and the short term drift of the laser power (1.25%) typical for our experiment.
Also here, the experimental data points match the simulated prediction very well.
As expected all evaluation methods overlap for the slowest thermal change protocol. This is consistent with the statement that the slowest ramp operates in the quasi-static limit (section 2.1). For faster thermal driving the quasi-static and linear response method show a significant deviation from theory.
Overall we see in figure 20 that the WSE equality provides the only valid

evaluation method for the time-scales on which our experimental setup operates.

4.2. Thermal and Mechanical Change

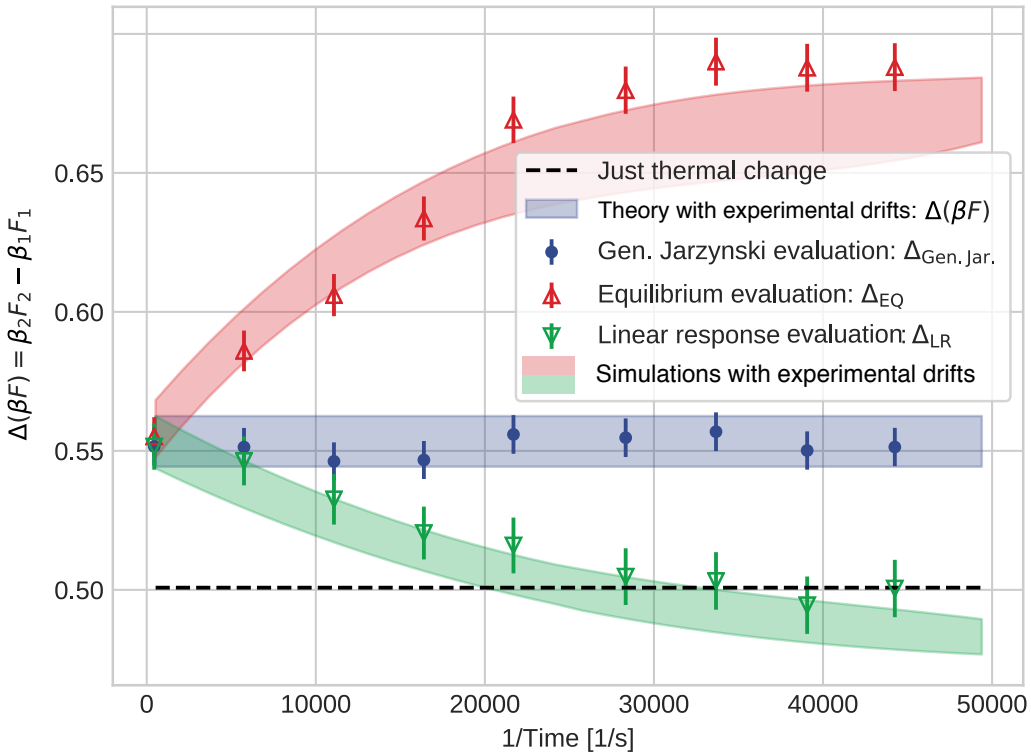

Figure 21: Comparison between experiment evaluation methods and theory for simultaneous mechanical and thermal driving: The blue data shows the actual theory predicted values compared to the blue data points which are based on the generalized Jarzynski evaluation. The red and the green rely on the equilibrium and the linear response method and depict a strong deviation from theory once we go to faster driving times. The black dashed line represents the theory predicted value for a state difference just given by the 65% thermal change and ignoring the 11% mechanical drive in the evaluation. For a detailed discussion of the methods, please take a look at the residual text of this subchapter.

In the following we will present the results for the simultaneous mechanical and thermal driving experiment as presented in section 3.3. We compare the estimated difference in normalised free energy evaluated by three different methods: the generalized Jarzynski equality, an evaluation assuming an equilibrium change and linear response theory.
The plot 21 is of the exact same design as the previous one showing the thermal only protocols results in figure 20 and therefore we again have an

inverse time axis as an x-axis and a normalized with respect to the different heat baths free energy difference on the y-scale.
The points in blue are evaluated with the generalized Jarzynski equality, see equation (2.12) on page 15, and lie within the theory prediction given by the partition function framework (blue area). The theory considers the overall drifts of the experimental setup over 5 hours measurement time, like changes in the cooling efficiency induced by pressure drifts inside the vacuum chamber. The error bars of the generalized Jarzynski evaluation were devised using the known experimental errors, like short term laser power fluctuation (1.25%), as well as taking into account the statistical measures extracted from the ensemble comprised of 15000 individual trajectories. As expected, the generalized Jarzynski fluctuation theorem shows no dependence on the speed of the protocols. The values also match exactly the theory (blue area). The blue area is not a fit, but is determined from independent experimental parameters.
The other two evaluation methods correspond to the equilibrium method (in red), equation (2.17), and the linear response theory (in green), equation (2.18). They match the red and green area well. The areas represent predictions for the expected experimental outcome of the two methods, respectively. The range covers the bounds of the predictions considering, as before, the drift of the cooling factors. These predictions are based on simulations of the physical system. The error bars were calculated analogously to the blue data error bars discussed in the above paragraph. As expected, the values agree with the generalized Jarzynski evaluation and with the theoretically expected value for the slowest protocol. This is consistent with the statement that here a quasi-static protocol is present (section 2.1). However, one sees significant deviations becoming larger with faster protocols and both methods (in red and green) fail to give a correct estimate of the normalized free energy difference. For the other evaluation methods we simulated an expectation which fits well with the experimental data.
In order to make sure that the mechanical change which happens simultaneous to the thermal one is of significance and cannot be neglected, we plotted an additional line to see where the normalised free energy difference would lie if we did not take the mechanical change into account when evaluating the partition function theory. We see that the black dashed line is located well outside of our error margins and therefore the 11% change in the particle's frequency makes up for a non-negligible contribution to the overall normalised thermodynamic free energy difference.
All things considered, we can see that for the time scales on which our experiment runs only the generalized Jarzynski evaluation is sufficient to experimentally determine the normalised free energy difference.

5. Conclusion

In the last chapter of this Master thesis I would like to take the opportunity to briefly highlight what particular insight was gained from the experiment and this thesis in general. Furthermore I give an outlook for the experimentally investigated techniques.
To implement the thermal and mechanical driving protocols we built on the already existing optical trapping setup using a hollow-core photonic crystal fibre (HCPCF) by Grass et al. [38, 48, 49]. Realizing the thermodynamic protocols in an optically levitated system is a convenient choice because of the easy control of the system's susceptibility via manipulating the trap stiffness and feedback cooling parameters. Hence, this setup allowed us to introduce thermal changes with a very high control to the centre of mass motion by using the velocity feedback scheme.
We implemented a control capability for the gain of the feedback system via an external signal source for this thesis. The HCPCF trapping setup was extended to feature a laser power control ability. This allowed us to engineer a stiffness change of the harmonic trapping potential, which was guided by a function generator. In particular we implemented a simultaneous mechanical and thermal drive of the trapped sub micron sized silica by just tuning the overall laser power knob of the optical trap.
We devised an experimental calibration and models to get a thorough understanding of the dynamics of the system and the bath during the experimental protocols. Especially, we paid close attention to the relation between the optical power change and the velocity feedback system.
It has been shown that the power of the optical trap can induce additional heating due to the absorption of laser light by the trapped nanoparticle [55]. This plays an important role since in addition our heat bath created by the velocity feedback control and the interaction with the surrounding gas molecules we would see an additional bath with the temperature depending on the laser power. Our cross check of the absorption model showed that we do not have a second bath. We conclude tthe nanoparticle equilibrates to the actual room temperature, when the feedback system is switched off.
Evaluating thermodynamic processes on a microscopic level and especially in a regime were we are away from equilibrium requires a lot of trajectories. Therefore, we developed a set of computational tools for evaluating the calibration data and to analyse the numerous trajectories for the fluctuation theorems using parallelising programming techniques (appendix section C). The added value to the computational toolbox will make it easier in the future to study different thermodynamic processes combining mechanical and thermal driving of the system of interest.
Simultaneous mechanical and thermal manipulation are essential steps in cyclic processes happening in heat engines [56]. But similar to the variety of applications of the Jarzynski equality [19] ranging from computational

physics over material science [31] to drug design [35] and nuclide acid folding [34], we hope that the now generalized Jarzynski fluctuation theorem, which accounts for both thermal and mechanical changes, will have a similar broad impact and open new understandings of thermal driving induced free energy changes in all kinds of systems.

We demonstrated very good agreement between the evaluated fluctuation theorems and the theoretical predictions for a simultaneous thermal and mechanical driving agent present. To our knowledge the presented experiments mark the first experimental investigation for the Williams-Searles-Evans equality [20] and the generalized Jarzynski equality [21, 22].

References

[1] R. Clausius, "Ueber eine veränderte Form des zweiten Hauptsatzes der mechanischen Wärmetheorie", de, Annalen der Physik und Chemie **169**, 481–506 (1854) (cit. on p. 1).

[2] G. Amontons, "Moyen de substituer commodement l'action du feu, a la force des hommes et des cheveaux pour mouvoir les machines (Method of substituting the force of fire for horse and man power to move machines)", Histoire de l'Académie royale des sciences, 112–126 (1699) (cit. on p. 1).

[3] R. Clausius, and W. Browne, *The Mechanical Theory of Heat* (Macmillan, 1879) (cit. on p. 1).

[4] W. Kelvin, J. Larmor, and J. Joule, "On an Absolute Thermometric Scale Founded on Carnot's Theory", Mathematical and Physical Papers **1**, 232 (1882) (cit. on p. 1).

[5] J. B. West, "Robert Boyle's landmark book of 1660 with the first experiments on rarified air", en, Journal of Applied Physiology **98**, 31–39 (2004) (cit. on p. 1).

[6] M. Crosland, "The origins of Gay-Lussac's law of combining volumes of gases", en, Annals of Science **17**, 1–26 (1961) (cit. on p. 1).

[7] C. Spurgin, "Gay-Lussac's gas-expansivity experiments and the traditional mis-teaching of 'Charles's Law'", en, Annals of Science **44**, 489–505 (1987) (cit. on p. 1).

[8] J. C. Maxwell, *Theory of heat*, 3d ed (Greenwood Press, Westport, Conn, 1970) (cit. on p. 1).

[9] D. Chandler, *Introduction to modern statistical mechanics* (Oxford University Press, New York, 1987) (cit. on pp. 1, 3, 26, 38, 54).

[10] W. Grimus, *Einführung in die Statistische Physik und Thermodynamik: Grundlagen und Anwendungen*, ger (Oldenbourg, München, 2010) (cit. on pp. 1, 9, 38, 53).

[11] J. W. Gibbs, *Elementary Principles in Statistical Mechanics* (Charles Scribner's Sons, New York, Mar. 1902) (cit. on p. 2).

[12] L. Onsager, "Reciprocal Relations in Irreversible Processes. I.", en, Physical Review **37**, 405–426 (1931) (cit. on pp. 2, 3).

[13] M. S. Green, "Markoff Random Processes and the Statistical Mechanics of Time-Dependent Phenomena. II. Irreversible Processes in Fluids", en, The Journal of Chemical Physics **22**, 398 (1954) (cit. on p. 3).

[14] R. Kubo, "Statistical-Mechanical Theory of Irreversible Processes. I. General Theory and Simple Applications to Magnetic and Conduction Problems", en, Journal of the Physical Society of Japan **12**, 570–586 (1957) (cit. on p. 3).

[15] H. B. Callen, and T. A. Welton, "Irreversibility and Generalized Noise", en, Physical Review **83**, 34–40 (1951) (cit. on p. 3).

[16] H. Nyquist, "Thermal Agitation of Electric Charge in Conductors", en, Physical Review **32**, 110–113 (1928) (cit. on p. 3).

[17] D. J. Evans, E. G. D. Cohen, and G. P. Morriss, "Probability of second law violations in shearing steady states", en, Physical Review Letters **71**, 2401–2404 (1993) (cit. on pp. 3, 12).

[18] D. J. Evans, and D. J. Searles, "Equilibrium microstates which generate second law violating steady states", en, Physical Review E **50**, 1645–1648 (1994) (cit. on p. 3).

[19] C. Jarzynski, "Nonequilibrium Equality for Free Energy Differences", en, Physical Review Letters **78**, 2690–2693 (1997) (cit. on pp. 3, 5–8, 11, 13–17, 32, 40, 43–46).

[20] S. R. Williams, D. J. Searles, and D. J. Evans, "Nonequilibrium Free-Energy Relations for Thermal Changes", Phys. Rev. Lett. **100**, 250601 (2008) (cit. on pp. 3, 5, 7, 11, 12, 14, 46, 54).

[21] R. Chelli, "Nonequilibrium work relations for systems subject to mechanical and thermal changes", en, The Journal of Chemical Physics **130**, 054102 (2009) (cit. on pp. 3, 5, 7, 11, 15, 46).

[22] R. Chelli, S. Marsili, A. Barducci, and P. Procacci, "Generalization of the Jarzynski and Crooks nonequilibrium work theorems in molecular dynamics simulations", Phys. Rev. E **75**, 050101 (2007) (cit. on pp. 3, 5, 7, 11, 15, 46).

[23] G. M. Wang, E. M. Sevick, E. Mittag, D. J. Searles, and D. J. Evans, "Experimental Demonstration of Violations of the Second Law of Thermodynamics for Small Systems and Short Time Scales", Physical Review Letters **89**, 050601 (2002) (cit. on p. 4).

[24] C. Jarzynski, "Equalities and Inequalities: Irreversibility and the Second Law of Thermodynamics at the Nanoscale", en, Annual Review of Condensed Matter Physics **2**, 329–351 (2011) (cit. on p. 4).

[25] U. Seifert, "Stochastic thermodynamics: principles and perspectives", The European Physical Journal B **64**, 423–431 (2008) (cit. on p. 4).

[26] D. M. Carberry, J. C. Reid, G. M. Wang, E. M. Sevick, D. J. Searles, and D. J. Evans, "Fluctuations and Irreversibility: An Experimental Demonstration of a Second-Law-Like Theorem Using a Colloidal Particle Held in an Optical Trap", en, Physical Review Letters **92** (2004) 10.1103/PhysRevLett.92.140601 (cit. on p. 4).

[27] G. Radons, B. Rumpf, and H. G. Schuster, eds., *Nonlinear dynamics of nanosystems*, OCLC: ocn463642396 (Wiley-VCH, Weinheim, 2010) (cit. on p. 4).

[28] U. Seifert, "Stochastic thermodynamics, fluctuation theorems and molecular machines", Reports on Progress in Physics **75**, 126001 (2012) (cit. on p. 5).

[29] A. Alemany, A. Mossa, I. Junier, and F. Ritort, "Experimental free-energy measurements of kinetic molecular states using fluctuation theorems", Nature Physics **8**, 688–694 (2012) (cit. on p. 6).

[30] F. Latinwo, K.-W. Hsiao, and C. M. Schroeder, "Nonequilibrium thermodynamics of dilute polymer solutions in flow", The Journal of Chemical Physics **141**, 174903 (2014) (cit. on p. 6).

[31] R. Demuynck, S. M. J. Rogge, L. Vanduyfhuys, J. Wieme, M. Waroquier, and V. Van Speybroeck, "Efficient Construction of Free Energy Profiles of Breathing Metal–Organic Frameworks Using Advanced Molecular Dynamics Simulations", Journal of Chemical Theory and Computation **13**, 5861–5873 (2017) (cit. on pp. 6, 46).

[32] G. Hummer, and A. Szabo, "Free energy reconstruction from nonequilibrium single-molecule pulling experiments", Proceedings of the National Academy of Sciences **98**, 3658–3661 (2001) (cit. on p. 6).

[33] D. Collin, F. Ritort, C. Jarzynski, S. B. Smith, I. Tinoco Jr, and C. Bustamante, "Verification of the Crooks fluctuation theorem and recovery of RNA folding free energies", Nature **437**, 231 (2005) (cit. on p. 6).

[34] A. N. Gupta, A. Vincent, K. Neupane, H. Yu, F. Wang, and M. T. Woodside, "Experimental validation of free-energy-landscape reconstruction from non-equilibrium single-molecule force spectroscopy measurements", Nat Phys **7**, 631–634 (2011) (cit. on pp. 6, 46).

[35] B. J. Williams-Noonan, E. Yuriev, and D. K. Chalmers, "Free Energy Methods in Drug Design: Prospects of "Alchemical Perturbation" in Medicinal Chemistry", Journal of Medicinal Chemistry **61**, 638–649 (2018) (cit. on pp. 6, 46).

[36] S. Ciliberto, "Experiments in Stochastic Thermodynamics: Short History and Perspectives", Phys. Rev. X **7**, 021051 (2017) (cit. on p. 6).

[37] M. Campisi, P. Hänggi, and P. Talkner, "Colloquium: Quantum fluctuation relations: Foundations and applications", Rev. Mod. Phys. **83**, 771–791 (2011) (cit. on p. 6).

[38] D. Grass, J. Fesel, S. G. Hofer, N. Kiesel, and M. Aspelmeyer, "Optical trapping and control of nanoparticles inside evacuated hollow core photonic crystal fibers", Applied Physics Letters **108**, 221103 (2016) (cit. on pp. 7, 18, 45).

[39] N. Kiesel, F. Blaser, U. Delic, D. Grass, R. Kaltenbaek, and M. Aspelmeyer, "Cavity cooling of an optically levitated submicron particle", Proceedings of the National Academy of Sciences **110**, 14180–14185 (2013) (cit. on pp. 7, 31).

[40] P. Asenbaum, S. Kuhn, S. Nimmrichter, U. Sezer, and M. Arndt, "Cavity cooling of free silicon nanoparticles in high vacuum", Nature Communications **4** (2013) 10.1038/ncomms3743 (cit. on p. 7).

[41] J. Gieseler, R. Quidant, C. Dellago, and L. Novotny, "Dynamic relaxation of a levitated nanoparticle from a non-equilibrium steady state", Nat Nano **9**, 358–364 (2014) (cit. on p. 7).

[42] A. Dechant, N. Kiesel, and E. Lutz, "All-Optical Nanomechanical Heat Engine", Phys. Rev. Lett. **114**, 183602 (2015) (cit. on p. 7).

[43] T. M. Hoang, R. Pan, J. Ahn, J. Bang, H. T. Quan, and T. Li, "Experimental Test of the Differential Fluctuation Theorem and a Generalized Jarzynski Equality for Arbitrary Initial States", Phys. Rev. Lett. **120**, 080602 (2018) (cit. on p. 7).

[44] A. Imparato, L. Peliti, G. Pesce, G. Rusciano, and A. Sasso, "Work and heat probability distribution of an optically driven Brownian particle: Theory and experiments", en, Physical Review E **76** (2007) 10.1103/PhysRevE.76.050101 (cit. on p. 7).

[45] L. D. Landau, E. M. Lifšic, L. P. Pitaevskij, and L. D. Landau, *Statistical physics, Part 1*, 3. ed, Course of theoretical physics (Elsevier, Amsterdam, 1980) (cit. on pp. 9, 53).

[46] S. Ciliberto, R. Gomez-Solano, and A. Petrosyan, "Fluctuations, Linear Response, and Currents in Out-of-Equilibrium Systems", Annual Review of Condensed Matter Physics **4**, 235–261 (2013) (cit. on p. 10).

[47] M. Hübner, "Simulationen zur Williams-Searles-Evans-Gleichung in harmonischen Potentialen", German, Bachelor Thesis (Friedrich-Alexander-Universität, Erlangen-Nürnberg, July 2014) (cit. on pp. 11, 55).

[48] D. Grass, "Optical Trapping and Transport of Nanoparticles with Hollow Core Photonic Crystal Fibers", English, MA thesis (University of Vienna, Vienna, 2013) (cit. on pp. 18, 19, 45).

[49] D. Grass, "Levitated optomechanics in vacuum using hollow core photonic crystal fibers and optical cavities", Doctoral Thesis (University of Vienna, Vienna, 2018) (cit. on pp. 18, 21, 45, 54).

[50] Y. Harada, and T. Asakura, "Radiation forces on a dielectric sphere in the Rayleigh scattering regime", en, Optics Communications **124**, 529–541 (1996) (cit. on p. 19).

[51] P. Langevin, "Sur la théorie du mouvement brownien", CR Acad. Sci. Paris **146**, 530 (1908) (cit. on pp. 19, 22, 55).

[52] J. Gieseler, B. Deutsch, R. Quidant, and L. Novotny, "Subkelvin Parametric Feedback Cooling of a Laser-Trapped Nanoparticle", Physical Review Letters **109**, 103603 (2012) (cit. on p. 20).

[53] A. Einstein, "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen", de, Annalen der Physik **322**, 549–560 (1905) (cit. on pp. 20, 55).

[54] M. Aspelmeyer, T. J. Kippenberg, and F. Marquardt, eds., *Cavity Optomechanics* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014) (cit. on p. 20).

[55] J. Millen, T. Deesuwan., P. Barker, and J. Anders, "Nanoscale temperature measurements using non-equilibrium Brownian dynamics of a levitated nanosphere", Nat Nano **9**, 425–429 (2014) (cit. on pp. 37, 45).

[56] J. Gieseler, and J. Millen, "Levitated Nanoparticles for Microscopic Thermodynamics—A Review", Entropy **20** (2018) 10.3390/e20050326 (cit. on p. 45).

[57] M. Rademacher, "Experimental implementation of the Williams-Searles-Evans equality", Bachelor Thesis (University of Vienna, Vienna, 2016) (cit. on p. 55).

[58] W. Sutherland, "LXXV. *A dynamical theory of diffusion for non-electrolytes and the molecular mass of albumin*", en, Philosophical Magazine Series 6 **9**, 781–785 (1905) (cit. on p. 55).

[59] M. von Smoluchowski, "Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen", de, Annalen der Physik **326**, 756–780 (1906) (cit. on p. 55).

[60] F. Cajori, *A history of mathematical notations* (Dover Publications, New York, 1993) (cit. on p. 56).

[61] J. Stewart, *Calculus: early transcendentals*, 6th ed (Thomson Brooks/Cole, Belmont, CA, 2008) (cit. on p. 56).

[62] J. C. Maxwell, and W. D. Niven, *The scientific papers of James Clerk Maxwell*, Dover phoenix editions (Dover Publications, Mineola, N.Y, 2003) (cit. on p. 57).

[63] L. Rayleigh, "VI. *The law of partition of kinetic energy*", en, Philosophical Magazine Series 5 **49**, 98–118 (1900) (cit. on p. 57).

[64] A. Setter, and M. Rademacher, "AshleySetter/optoanalysis: v4.0.1 Release", (2017) 10.5281/zenodo.1042526 (cit. on p. 59).

[65] L. Dagum, and R. Menon, "OpenMP: an industry standard API for shared-memory programming", IEEE Computational Science and Engineering **5**, 46–55 (1998) (cit. on p. 74).

[66] B. Chapman, G. Jost, and R. v. d. Pas, *Using OpenMP: portable shared memory parallel programming*, Scientific and engineering computation, OCLC: ocn145944336 (MIT Press, Cambridge, Mass, 2008) (cit. on p. 74).

Appendices

A. Additional calculations

A.1. Free energy difference given by the partition function

The partition function theory provides us with the following definition of thermodynamic free energy F [45]:

$$F = \frac{1}{\beta} \ln(Z) \tag{A.1}$$

In relation (A.1) β is the inverse temperature, i.e., $1/(k_B T)$, and Z represents the partition function. By using the definition of the free energy via the partition function as can be seen in equation (A.1), we can calculate the expected normalised free energy difference for a given mechanical and thermal state difference as follows in relation (A.2). By having a normalised free energy difference, we mean that the free energies $F_{1,2}$ in the difference expression are normalised in respect to their environmental heat bath temperatures $\beta_{1,2}$.

$$\beta_2 F_2 - \beta_1 F_1 = \ln(Z_2) - \ln(Z_1) \tag{A.2}$$

The respective partition functions $Z_{1,2}$ of the beginning and the end state of a thermodynamic process involving a thermal and a mechanical change can be expressed in the following way:

$$Z_{1,2} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \exp\left(-\beta_{1,2}\left(\lambda_{1,2}\frac{k}{2}x^2 + \frac{p^2}{2m}\right)\right) dxdp \tag{A.3}$$

In equation (A.3) $\lambda_{1,2}$ is the mechanical change parameter modulating the spring constant k which in our optical trapping setup is the stiffness of the laser trap.
After plugging in the respective partition function seen in relation (A.3) into the free energy difference expression in (A.2), integrating and using some logarithmic relationships, we finally arrive at the end result seen in equation (A.4) describing the normalised free energy difference of a system undergoing a thermal and a mechanical state change.

$$\beta_2 F_2 - \beta_1 F_1 = \ln\left(\frac{\beta_2}{\beta_1}\right) + \frac{1}{2}\ln\left(\frac{\lambda_2}{\lambda_1}\right) \tag{A.4}$$

For further reference to this calculation, I would like to refer the reader to publication [10, 45].

A.2. Relation of the linear response theory to WSE fluctuation theorem

By starting out with the WSE equality [20] and making a first order taylor expansion of the exponential and the following logarithm function, we will see how to recover the linear response expression in calculation (A.5).

$$\begin{aligned}
\beta_2 F_2 - \beta_1 F_1 &= -\ln\left(\left\langle e^{-\Theta}\right\rangle\right) \qquad \text{with} \qquad \Theta = \int_0^\tau \dot{\beta}(t) H_0(\Gamma(t)) dt \\
&\approx -\log\left(\left\langle 1 - \Theta + \frac{\Theta^2}{2}\right\rangle\right) \\
&\approx \langle\Theta\rangle - \frac{\langle\Theta^2\rangle}{2} + \frac{\langle\Theta\rangle^2}{2} \\
&= \langle\Theta\rangle - \frac{1}{2}\left\langle\Theta^2 - \langle\Theta\rangle^2\right\rangle
\end{aligned} \tag{A.5}$$

A.3. Effective temperature of a feedback cooled levitated nanoparticle

The following derivation for the effective centre of mass temperature of a velocity feedback cooled levitated nanoparticle is based upon the one for an effective mode temperature done by Grass in his doctoral thesis [49].
Let us start by looking at the PSD S_{xx} of a levitated nanoparticle undergoing velocity feedback cooling:

$$S_{xx}(\omega) = \frac{1}{\pi m} \frac{2k_B T_0 \gamma_\mathrm{p}}{(\omega^2 - \Omega^2) + \omega^2 \gamma_\mathrm{eff}^2} \tag{A.6}$$

We then integrate over the entire PSD, equation (A.6), to get the positional variance of our oscillator:

$$\langle x^2\rangle = \int_0^\infty S_{xx}(\omega) d\omega = \frac{k_B T_0 \gamma_\mathrm{p}}{m\Omega^2 \gamma_\mathrm{eff}} \tag{A.7}$$

Relating the result in equation (A.7) to the equipartition theorem for a cooled particle [9], which states:

$$\langle x^2\rangle = \frac{k_B T_\mathrm{eff}}{m\Omega^2} \tag{A.8}$$

allows us to express an effective temperature for a velocity feedback cooled particle:

$$T_{\text{eff}} = \frac{m\Omega^2}{k_B\langle x^2\rangle} = T_0\frac{\gamma_{\text{p}}}{\gamma_{\text{eff}}} \tag{A.9}$$

Since we now that $\gamma_{\text{eff}} = \gamma_{\text{p}} + \gamma_{\text{fb}}$, we arrive at the final expression for the effective temperature T_{eff} of a feedback cooled nanoparticle:

$$T_{\text{eff}} = T_0 \cdot \frac{\gamma_{\text{p}}}{\gamma_{\text{p}} + \gamma_{\text{fb}}} \tag{A.10}$$

A.4. Transformation of the Langevin equation starting from unitless quantities

The following section is directly taken from a previous Bachelor thesis [57] by the author and just put into the appendix of this Master thesis for completeness.
For devising the optimal driving protocols for our system of interest we are going to make use of a computer simulation, which in its original form was coded by Moritz Hübner for investigating an overdamped system. As this was coded for a Bachelor thesis [47], which focused purely on the theory part of the problem, all the units and physical constants are not equalling the ones used in experiments in order to guarantee a high numerical precision and a minimum of computation time.
The goal of the simulation is to obtain the optimal driving protocol for the feedback cooling of the optically trapped sub-micron particle and therefore we have to find a way to make use of the unitless quantities delivered by the numerical simulation. Thus, we want to analyse the relation between the unitless properties of the computer simulation and real world experimental values in the SI-System and ultimately come up with a simple transformation formula between the systems. Our general Langevin-equation [51] that is used to describe the particle equals:

$$\ddot{x}_s + \gamma_s \cdot \dot{x}_s + \frac{k_s}{m_s} \cdot x_s = \sqrt{2 \cdot D_s} = \sqrt{2 \cdot \frac{\gamma_s}{m_s \cdot \beta_s}} \cdot \xi_{t_s} \tag{A.11}$$

In equation (A.11) D_s represents the diffusion constant on the simulation's scale, which follows from the Einstein–Smoluchowski relation [53, 58, 59]. γ_s is the simulation's dimensionless friction constant, β_s the thermodynamic beta of the simulation, k_s the spring constant given in dimensions of the simulation and m_s the mass of the particle in the simulation's system. ξ_{t_s} is the correlation function describing the characteristics of the thermal noise in the simulation's dimensions. To point out the difference of the quantities in the simulation's system to the experimental system, we switch from

Newton's dot notation for differentiation [60] to the Leibniz's notation [61]. Furthermore we know that $\frac{k_s}{m_s} = \omega_s^2$, so equation (A.11) looks like:

$$\frac{d^2x_s}{dt_s^2} + \gamma_s \cdot \frac{dx_s}{dt_s} + \omega_s^2 \cdot x_s = \sqrt{2 \cdot \frac{\gamma_s}{m_s \cdot \beta_s}} \cdot \xi_{t_s} \tag{A.12}$$

As we are allowed to write $\frac{dx_s}{dt_s}$ as $\frac{dx_s}{dt} \cdot \frac{dt}{dt_s}$, we have:

$$\frac{d^2x_s}{dt^2} \cdot \left(\frac{dt}{dt_s}\right)^2 + \gamma_s \cdot \frac{dx_s}{dt} \cdot \frac{dt}{dt_s} + \omega_s^2 \cdot x_s = \sqrt{2 \cdot \frac{\gamma_s}{m_s \cdot \beta_s}} \cdot \xi_t \cdot K_{t_s,t} \tag{A.13}$$

In equation (A.13) $K_{t_s,t}$ represents a constant that enables us to transform the correlation function ξ from our t_s-system to the t-system. We make the ansatz that $\frac{dt}{dt_s} = \frac{\omega_s}{\omega}$ and divide equation (A.13) by $\left(\frac{dt}{dt_s}\right)^2$, we get:

$$\frac{d^2x_s}{dt^2} + \gamma_s \cdot \frac{\omega}{\omega_s} \cdot \frac{dx_s}{dt} + \omega^2 \cdot x_s = \left(\frac{\omega}{\omega_s}\right)^2 \cdot \sqrt{2 \cdot \frac{\gamma_s}{m_s \cdot \beta_s}} \cdot \xi_t \cdot K_{t_s,t} \tag{A.14}$$

Using relation (A.15) we have determined $K_{t_s,t}$.

$$\frac{d\xi_{t_s}}{d\xi_t} = \sqrt{\frac{dt}{dt_s}} = \sqrt{\frac{\omega_s}{\omega}} \stackrel{!}{=} \frac{\xi_{t_s}}{\xi_t} = K_{t_s,t} \tag{A.15}$$

Thus, we will use the following transformation for our time-scales:

$$t_s = \frac{\omega}{\omega_s} \cdot t \tag{A.16}$$

In the last step of the relation (A.15) we assume that the ξs are very very small and therefore we can set the proportion of the infinitesimal values to the one of the experimental values. By applying the relation $\frac{\omega_s}{\gamma_s} = Q = \frac{\omega}{\gamma}$, where Q stands for the mechanical quality factor of the system, equation (A.14) transforms to:

$$\frac{d^2x_s}{dt^2} + \gamma \cdot \frac{dx_s}{dt} + \omega^2 \cdot x_s = \frac{\omega^2}{\omega_s^2} \cdot \sqrt{2 \cdot \frac{\omega_s}{\omega} \cdot \frac{\gamma}{m_s \cdot \beta_s}} \cdot \xi_t \cdot K_{t_s,t} \tag{A.17}$$

using relation (A.15), one ends up with:

$$\frac{d^2x_s}{dt^2} + \gamma \cdot \frac{dx_s}{dt} + \omega^2 \cdot x_s = \frac{\omega}{\omega_s} \cdot \sqrt{2 \cdot \frac{\gamma}{m_s \cdot \beta_s}} \cdot \xi_t \tag{A.18}$$

We will now rescale our position scale via the ansatz that is written in equation (A.19).

$$x_s = \sqrt{\frac{m \cdot \omega^2 \cdot \beta}{m_s \cdot \omega_s^2 \cdot \beta_s}} \cdot x \tag{A.19}$$

and end up with:

$$\frac{d^2x}{dt^2} + \gamma \cdot \frac{dx}{dt} + \omega^2 \cdot x = \sqrt{2 \cdot \frac{\gamma}{m \cdot \beta}} \cdot \xi_t \tag{A.20}$$

The motivation for using ansatz (A.19) lies behind the equipartition theorem [62, 63]. As one is then able to set equal both system's mean squared positions, we have:

$$m_s \cdot \omega_s^2 \cdot \beta_s \cdot \left\langle x_s^2 \right\rangle = m \cdot \omega^2 \cdot \beta \cdot \left\langle x^2 \right\rangle \tag{A.21}$$

Now that we have acquired the transformations for the time and the position scales of our system, we can easily get to the transformation that gives us the energy coming from an unit-less energy, as we are also allowed to express $m = \frac{m}{m_s} \cdot m_s$. The newly found relation can be seen in (A.22).

$$F = \frac{x^2 \cdot m}{t^2} = \frac{\cancel{m_s} \cdot \cancel{\omega_s^2} \cdot \beta_s}{\cancel{m} \cdot \cancel{\omega^2} \cdot \beta} \cdot \frac{\cancel{\omega^2}}{\cancel{\omega_s^2}} \cdot \frac{\cancel{m}}{\cancel{m_s}} \cdot \underbrace{\frac{x_s^2 \cdot m_s}{t_s^2}}_{F_s} \tag{A.22}$$

$$F = \frac{\beta_s}{\beta} \cdot F_s \tag{A.23}$$

In equation (A.22) and (A.23) the variable F especially denotes the free energy of a system, however this transformation still remains universally applicable to all kinds of energies of course.

B. Table of different thermodynamic protocols

The following protocol times τ in the table 1 were used to generate the leading and the trailing edge times of the modified pulse signal used to control the feedback gain in the case of the thermal driving protocols and the trapping power in the case of the thermal and mechanical driving protocols. The overall control signal was constructed such that $\tau_{\text{equilibration}} = 12.5\text{ms} - \tau$ would yield the respective equilibration time $\tau_{\text{equilibration}}$ after cooling/trapping power was ramped up/down.

run#	1	2	3	4	5	6	7	8	9
$\tau(\mu\mathbf{s})$	22.6	2262.44	46.11	29.73	90.38	25.68	35.31	61.07	173.82

Table 1: Different executions times τ by run number (#) of the thermodynamic protocol used for the thermal change and the simultaneous thermal and mechanical change experiment

C. Software Library and Parallel Programming

The analysis of the calibration data, which was taken before and after the thermodynamic protocols in order to infer the temperature model and mechanical change model of our induced system change, was done in Python using the optoanalysis data analysis library written by Ashley Setter and the thesis' author, Markus Rademacher [64]. To cope with the many calibration files and the different evaluation techniques, we decided to use the Python's built-in multiprocessing library to speed things up. The final evaluation of the different evaluation techniques, which are depicted in figure 20 and 21, of the thermodynamic protocols with the calibration data deduced models, which were extracted by using the afore mentioned Python code, was done with a parallelised C++ script which was based in its original form on the performed unitless simulations, which determined the optimal driving protocols for the experimental system.
Therefore the first subchapter will show the Python code used on the calibration data and an accompanied comment to it explaining how we exactly made use of Python's multiprocessing internal library to guarantee a swift computation. The second subchapter will focus on the C++ program and will also describe the implemented methods to parallelise the script in a separate commentary.

C.1. Python code

C.1.1. Code commentary

The main feature of the code in the following subsection is the implementation of the internal Python multiprocessing library, which allows the script to harness all the cores of the computer's central processing unit instead of the usual single core execution. Therefore we can analyse multiple calibration time traces at the same time and reduce the computation time by more than an order of magnitude.
The way this is implemented is by writing a subscript, basically a separate custom function, which goes through the entire analysis that has to be done for a single file. In our case this is the `full_analysis` function starting in code line 95. The function loads the respective file, calculates the power spectrum density (PSD) of the time trace, fits the PSD, calculates variances, cooling factors and so on. The documentation for the functions called from the optoanalysis library is available under `http://optoanalysis.readthedocs.io`. The important line is at the end of this sub-function, because in order to use the internal multiprocessing unit of Python we are allowed to only have just one element in the return statement of this function. This return statement can be seen in code line 268. We are still able to get all of the necessary different parameters out of the analysis,

because we due a nifty little trick in creating a single python list element, basically this is like a vector or an array, in which we save all of these parameters calculated off a single time trace. Following this we put this single list of convoluted results into the return statement and by doing so fulfil the condition of effectively only returning one variable.
The actual code line parallelising this analysis process can be seen in code line 273. This multiprocessing function takes care of everything starting from assigning different computation workers to the different CPU threads, to allocating the correct memory in the RAM and closing the workers after the analysis of the single file has finished in order to reallocate the computational thread and free up vital RAM space. After the multiprocessing function worked its way through the given list of different file names it return a list of the returned results of the called sub function. Since each item on the file name list gets send to the before spoken sub script, which returns a list of characteristic parameters, we end up with a list of lists of these characteristic parameters. That is why we then have to sort this nested list structure with a loop again in code line 281 in order to have an easier variable calling structure. This last sorting bit is just the author's preference how to deal with the nested lists and not required at all for further processing.
The code finishes with the plotting commands to create parts of the visual data analysis results shown in section 3.3.

Code

```
import matplotlib as mpl
mpl.use('Agg')
import sys
sys.settrace
import numpy as np
import optoanalysis
import matplotlib.pyplot as plt
import glob
import os
from multiprocessing import Pool, cpu_count
from matplotlib import rc
from optoanalysis import take_closest

#import gc
mpl.rcParams.update({'figure.max_open_warning': 0})
print(mpl.matplotlib_fname())
mpl.interactive(False)

#print(gc.isenabled())
plt.rc('font', family='sans-serif')
plt.rc('font', serif='Helvetica')
plt.rcParams['font.size'] = 16

os.chdir('/media/fastRaid0/people/markus/20180625 - WSE/')
saving_dir = os.getcwd()

if not os.path.exists('PSDs'):
    os.makedirs('PSDs')
if not os.path.exists('Fits'):
    os.makedirs('Fits')
if not os.path.exists('FitsAuto'):
    os.makedirs('FitsAuto')

moglabs_voltages = list(range(31))
TrappingPowers = np.array([ 859.88265609,  856.8535346 ,  858.33283325,  860, 855.54388978,
853.28898773,  855.93485505,  853.64726942, 849.67232115,  845.41227187,  847.12248954,
836.3429235 ,844.17833954,  834.66166147,  830.24682401,  827.0589614, 819.7999764 ,
811.00675601,  810.12235012,  805.83314546,805.4805195 ,  800.30815629,  800.2622096 ,  798.00,
789.13400023,  776.77851303,  773.00416036,  763.44434409,758.71153396,  748.73765173,
746.01047455])
trapping_powers = np.linspace(min(TrappingPowers), max(TrappingPowers), num=31)[::-1]
peakFreqs = np.empty(len(moglabs_voltages))
PSDs = []
uncooledFits = []
uncooledFittedAreas = []
uncooledFittedAreasErr = []
uncooledFittedGammas = []
uncooledFittedGammasUfloat = []
uncooledFittedOmegas = []
uncooledFittedOmegasErr = []
uncooledFittedOmegasFileset_0 = []
uncooledFittedOmegasFileset_1 = []
uncooledFittedOmegasFileset_2 = []
uncooledFittedOmegasFileset_3 = []
cooledFits = []
cooledFittedAreas = []
cooledFittedAreasErr = []
cooledFittedGammas = []
cooledFittedGammasUfloat = []
cooledFittedOmegas = []
cooledFittedOmegasErr = []
cooledFittedOmegasFileset_0 = []
cooledFittedOmegasFileset_1 = []
cooledFittedOmegasFileset_2 = []
cooledFittedOmegasFileset_3 = []
uncooledRawAreas = []
cooledRawAreas = []
coolingfactor_var = []
coolingfactor_var_big_filter = []
Gamma_by_autocorrelation = []
```

```
Mean_Of_Variances = []
Var_Of_Variances = []
Mean_Of_Variances_cooled = []
pressureList =[]
Gamma_by_autocorrelation_cooled = []
Var_cooled_BigFilter = []

#For manually optimized fitting:

FittingParamsTable = optoanalysis.ORGTableData(glob.glob('*.org')[0])

def get_fitting_params(voltage,ColumnName):
    return
eval(np.array(FittingParamsTable.ORGTableData[FittingParamsTable.ORGTableData.RunNo ==
str(voltage)][ColumnName])[0])

def find_peak(data,lowerFreq=330e3,upperFreq=420e3):
    lowerIndex = np.where(data.freqs == take_closest(data.freqs,lowerFreq))[0][0]
    upperIndex = np.where(data.freqs == take_closest(data.freqs, upperFreq))[0][0]
    MaxPSD = max(data.PSD[lowerIndex:upperIndex])
    centralIndex = np.where(data.PSD == MaxPSD)[0][0]
    '''
    if data.filename[-13:-11] == ' 3':
        return 420000
    elif data.filename[-13:-11] in ['16','17','20','26']:
        return 400000
    else:
        return data.freqs[centralIndex]
    '''
    return data.freqs[centralIndex]

def full_analysis(index):
    omega_ref=476052
    '''if(index==4):
        PointsToLoadCooled=int(.2E7)
    else:
        PointsToLoadCooled=int(.02E7) #.02E7
    '''
    PointsToLoadCooled=-1#int(4e5) #int(5E5) #int(.1E7) #.02E7
    PointsToLoad=-1#int(4e5) #int(5E5) #int(.1E7)
    NPerSegmentPSD=1e6 #1e5
    MonitorChannel = 1
    Normalisation = True
    cooled = optoanalysis.load_data(sorted(glob.glob('*- %imV - tt.dat'%
(moglabs_voltages[index]))))[0],RelativeChannelNo=MonitorChannel,
SampleFreq=10E6,PointsToLoad=PointsToLoadCooled,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    uncooled = optoanalysis.load_data(sorted(glob.glob('*- %imV_noFB - tt.dat'%
(moglabs_voltages[index]))))[0],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoad,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    cooled_1 = optoanalysis.load_data(sorted(glob.glob('*- %imV - tt.dat'%
(moglabs_voltages[index]))))[1],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoadCooled,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    uncooled_1 = optoanalysis.load_data(sorted(glob.glob('*- %imV_noFB - tt.dat'%
(moglabs_voltages[index]))))[1],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoad,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    cooled_2 = optoanalysis.load_data(sorted(glob.glob('*- %imV - tt.dat'%
(moglabs_voltages[index]))))[2],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoadCooled,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    uncooled_2 = optoanalysis.load_data(sorted(glob.glob('*- %imV_noFB - tt.dat'%
(moglabs_voltages[index]))))[2],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoad,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    cooled_3 = optoanalysis.load_data(sorted(glob.glob('*- %imV - tt.dat'%
(moglabs_voltages[index]))))[3],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoadCooled,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
```

```
    uncooled_3 = optoanalysis.load_data(sorted(glob.glob('*- %imV_noFB - tt.dat'%
(moglabs_voltages[index])))[3],RelativeChannelNo=MonitorChannel, SampleFreq=10E6,
PointsToLoad=PointsToLoad,
ObjectType='thermo',NPerSegmentPSD=NPerSegmentPSD,NormaliseByMonitorOutput=Normalisation)
    #print(moglabs_voltages[index],1)
    FitParamsUncooled = np.array(get_fitting_params(moglabs_voltages[index],'dataset
noFB_0'))
    FitParamsUncooled[0]=find_peak(uncooled)
    #print(moglabs_voltages[index],2)
    fit_uncooled =
uncooled.get_fit_auto(CentralFreq=FitParamsUncooled[0],MinWidth=FitParamsUncooled[1],MaxWidth=Fit
 MakeFig=True, show_fig=False)
    #print(moglabs_voltages[index],3)
    FitParamsUncooled_1 = np.array(get_fitting_params(moglabs_voltages[index],'dataset
noFB_1'))
    FitParamsUncooled_1[0]=find_peak(uncooled_1)
    fit_uncooled_1 =
uncooled_1.get_fit_auto(CentralFreq=FitParamsUncooled_1[0],MinWidth=FitParamsUncooled_1[1],MaxWid
 MakeFig=True, show_fig=False)
    FitParamsUncooled_2 = np.array(get_fitting_params(moglabs_voltages[index],'dataset
noFB_2'))
    FitParamsUncooled_2[0]=find_peak(uncooled_2)
    fit_uncooled_2 =
uncooled_2.get_fit_auto(CentralFreq=FitParamsUncooled_2[0],MinWidth=FitParamsUncooled_2[1],MaxWid
 MakeFig=True, show_fig=False)
    FitParamsUncooled_3 = np.array(get_fitting_params(moglabs_voltages[index],'dataset
noFB_3'))
    FitParamsUncooled_3[0]=find_peak(uncooled_3)
    fit_uncooled_3 =
uncooled_3.get_fit_auto(CentralFreq=FitParamsUncooled_3[0],MinWidth=FitParamsUncooled_3[1],MaxWid
 MakeFig=True, show_fig=False)
    FitParamsCooled = np.array(get_fitting_params(moglabs_voltages[index],'dataset 0'))
    FitParamsCooled[0]=find_peak(uncooled)
    fit_cooled =
cooled.get_fit_auto(CentralFreq=FitParamsCooled[0],MinWidth=FitParamsCooled[1],MaxWidth=FitParams
 MakeFig=True, show_fig=False)
    FitParamsCooled_1 =  np.array(get_fitting_params(moglabs_voltages[index],'dataset 1'))
    FitParamsCooled_1[0]=find_peak(uncooled_1)
    fit_cooled_1 =
cooled_1.get_fit_auto(CentralFreq=FitParamsCooled_1[0],MinWidth=FitParamsCooled_1[1],MaxWidth=Fit
 MakeFig=True, show_fig=False)
    FitParamsCooled_2 =  np.array(get_fitting_params(moglabs_voltages[index],'dataset 2'))
    FitParamsCooled_2[0]=find_peak(uncooled_2)
    fit_cooled_2 =
cooled_2.get_fit_auto(CentralFreq=FitParamsCooled_2[0],MinWidth=FitParamsCooled_2[1],MaxWidth=Fit
 MakeFig=True, show_fig=False)
    FitParamsCooled_3 =  np.array(get_fitting_params(moglabs_voltages[index],'dataset 3'))
    FitParamsCooled_3[0]=find_peak(uncooled_3)
    fit_cooled_3 =
cooled_3.get_fit_auto(CentralFreq=FitParamsCooled_3[0],MinWidth=FitParamsCooled_3[1],MaxWidth=Fit
 MakeFig=True, show_fig=False)
    MultiPSD =
optoanalysis.multi_plot_PSD([uncooled,cooled,uncooled_1,cooled_1,uncooled_2,cooled_2,uncooled_3,c
['noFB','cooled','noFB__1','cooled_1','noFB_2','cooled_2','noFB_3','cooled_3'],alphaArray=
[.3,.3,.3,.3,.3,.3,.3,.3,.3,.3],show_fig=False)
    coolingfactor_calc_by_var =
(np.var(uncooled.filter_data(freq=fit_uncooled[0].n/(2*np.pi),PeakWidth=40e3, MakeFig=False,
show_fig=False)[1])/np.var(cooled.filter_data(freq=fit_cooled[0].n/(2*np.pi),PeakWidth=40e3,
MakeFig=False, show_fig=False)
[1]),np.var(uncooled_1.filter_data(freq=fit_uncooled_1[0].n/(2*np.pi),PeakWidth=40e3,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_1.filter_data(freq=fit_cooled_1[0].n/(2*np.pi),PeakWidth=40e3, MakeFig=False,
show_fig=False)
[1]),np.var(uncooled_2.filter_data(freq=fit_uncooled_2[0].n/(2*np.pi),PeakWidth=40e3,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_2.filter_data(freq=fit_cooled_2[0].n/(2*np.pi),PeakWidth=40e3, MakeFig=False,
show_fig=False)
[1]),np.var(uncooled_3.filter_data(freq=fit_uncooled_3[0].n/(2*np.pi),PeakWidth=40e3,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_3.filter_data(freq=fit_cooled_3[0].n/(2*np.pi),PeakWidth=40e3, MakeFig=False,
show_fig=False)[1]))
```

```
144
145     BigFilterCenterFreq = 365e3
146     BigFilterWidth = 90e3
147     coolingfactor_calc_by_var_big_filter = (
148         np.var(uncooled.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)
[1])/np.var(cooled.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth, MakeFig=False,
show_fig=False)[1]),
149         np.var(uncooled_1.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_1.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]),
150         np.var(uncooled_2.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_2.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]),
151         np.var(uncooled_3.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)
[1])/np.var(cooled_3.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]))
152     var_cooled_big_filter = (
153         np.var(cooled.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]),
154         np.var(cooled_1.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]),
155         np.var(cooled_2.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]),
156         np.var(cooled_3.filter_data(freq=BigFilterCenterFreq,PeakWidth=BigFilterWidth,
MakeFig=False, show_fig=False)[1]))
157
158     RawArea_uncooled = (
159
uncooled.calc_area_under_PSD(fit_uncooled[0].n/(2*np.pi)-40000,fit_uncooled[0].n/(2*np.pi)+40000)

160
uncooled_1.calc_area_under_PSD(fit_uncooled_1[0].n/(2*np.pi)-40000,fit_uncooled_1[0].n/(2*np.pi)+

161
uncooled_2.calc_area_under_PSD(fit_uncooled_2[0].n/(2*np.pi)-40000,fit_uncooled_2[0].n/(2*np.pi)+

162
uncooled_3.calc_area_under_PSD(fit_uncooled_3[0].n/(2*np.pi)-40000,fit_uncooled_3[0].n/(2*np.pi)+

163         )
164     RawArea_cooled = (
165
cooled.calc_area_under_PSD(fit_cooled[0].n/(2*np.pi)-40000,fit_cooled[0].n/(2*np.pi)+40000),
166
cooled_1.calc_area_under_PSD(fit_cooled_1[0].n/(2*np.pi)-40000,fit_cooled_1[0].n/(2*np.pi)+40000)

167
cooled_2.calc_area_under_PSD(fit_cooled_2[0].n/(2*np.pi)-40000,fit_cooled_2[0].n/(2*np.pi)+40000)

168
cooled_3.calc_area_under_PSD(fit_cooled_3[0].n/(2*np.pi)-40000,fit_cooled_3[0].n/(2*np.pi)+40000)

169         )
170
171     fit_uncooled[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_uncooled[0]/(2*np.pi),fit_uncooled[2]/(2*np.pi),fit_uncooled[1],RawArea_uncooled[0

172     fit_uncooled[3].savefig('Fits/fit_uncooled_%imV.pdf'%(moglabs_voltages[index]))
173     plt.close()
174     fit_uncooled_1[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_uncooled_1[0]/(2*np.pi),fit_uncooled_1[2]/(2*np.pi),fit_uncooled_1[1],RawArea_unco

175     fit_uncooled_1[3].savefig('Fits/fit_uncooled_%imV_1.pdf'%(moglabs_voltages[index]))
176     plt.close()
177     fit_uncooled_2[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_uncooled_2[0]/(2*np.pi),fit_uncooled_2[2]/(2*np.pi),fit_uncooled_2[1],RawArea_unco
```

```
178     fit_uncooled_2[3].savefig('Fits/fit_uncooled_%imV_2.pdf'%(moglabs_voltages[index]))
179     plt.close()
180     fit_uncooled_3[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_uncooled_3[0]/(2*np.pi),fit_uncooled_3[2]/(2*np.pi),fit_uncooled_3[1],RawArea_unco

181     fit_uncooled_3[3].savefig('Fits/fit_uncooled_%imV_3.pdf'%(moglabs_voltages[index]))
182     plt.close()
183     fit_cooled[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_cooled[0]/(2*np.pi),fit_cooled[2]/(2*np.pi),fit_cooled[1],RawArea_cooled[0]))
184     fit_cooled[3].savefig('Fits/fit_cooled_%imV.pdf'%(moglabs_voltages[index]))
185     plt.close()
186     fit_cooled_1[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_cooled_1[0]/(2*np.pi),fit_cooled_1[2]/(2*np.pi),fit_cooled_1[1],RawArea_cooled[1])

187     fit_cooled_1[3].savefig('Fits/fit_cooled_%imV_1.pdf'%(moglabs_voltages[index]))
188     plt.close()
189     fit_cooled_2[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_cooled_2[0]/(2*np.pi),fit_cooled_2[2]/(2*np.pi),fit_cooled_2[1],RawArea_cooled[2])

190     fit_cooled_2[3].savefig('Fits/fit_cooled_%imV_2.pdf'%(moglabs_voltages[index]))
191     plt.close()
192     fit_cooled_3[4].set_title('$\omega$={}Hz, $\Gamma$={}Hz,\n Area={}, RawArea=
{}'.format(fit_cooled_3[0]/(2*np.pi),fit_cooled_3[2]/(2*np.pi),fit_cooled_3[1],RawArea_cooled[3])

193     fit_cooled_3[3].savefig('Fits/fit_cooled_%imV_3.pdf'%(moglabs_voltages[index]))
194     plt.close()
195
196     GammaAuto =
uncooled.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,MakeFig=True,show_f

197
GammaAuto_1=uncooled_1.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,MakeF

198
GammaAuto_2=uncooled_2.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,MakeF

199
GammaAuto_3=uncooled_3.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,MakeF

200
201     GammaAutoCooled =
cooled.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,MakeFig=True,show_fig

202
GammaAutoCooled_1=cooled_1.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,M

203
GammaAutoCooled_2=cooled_2.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,M

204
GammaAutoCooled_3=cooled_3.calc_gamma_from_variance_autocorrelation_fit(NumberOfOscillations=10,M

205
206     GammaAuto[1].savefig('FitsAuto/fit_uncooled_%imV.pdf'%(moglabs_voltages[index]))
207     plt.close()
208     GammaAuto_1[1].savefig('FitsAuto/fit_uncooled_%imV_1.pdf'%(moglabs_voltages[index]))
209     plt.close()
210     GammaAuto_2[1].savefig('FitsAuto/fit_uncooled_%imV_2.pdf'%(moglabs_voltages[index]))
211     plt.close()
212     GammaAuto_3[1].savefig('FitsAuto/fit_uncooled_%imV_3.pdf'%(moglabs_voltages[index]))
213     plt.close()
214     GammaAutoCooled[1].savefig('FitsAuto/fit_cooled_%imV.pdf'%(moglabs_voltages[index]))
215     plt.close()
216     GammaAutoCooled_1[1].savefig('FitsAuto/fit_cooled_%imV_1.pdf'%(moglabs_voltages[index]))
217     plt.close()
218     GammaAutoCooled_2[1].savefig('FitsAuto/fit_cooled_%imV_2.pdf'%(moglabs_voltages[index]))
219     plt.close()
220     GammaAutoCooled_3[1].savefig('FitsAuto/fit_cooled_%imV_3.pdf'%(moglabs_voltages[index]))
221     plt.close()
222
223     Gamma_by_autocorrelation = (
```

```
        GammaAuto[0].n/(2*np.pi),
        GammaAuto_1[0].n/(2*np.pi),
        GammaAuto_2[0].n/(2*np.pi),
        GammaAuto_3[0].n/(2*np.pi)
    )

    Gamma_by_autocorrelation_cooled = (
        GammaAutoCooled[0].n/(2*np.pi),
        GammaAutoCooled_1[0].n/(2*np.pi),
        GammaAutoCooled_2[0].n/(2*np.pi),
        GammaAutoCooled_3[0].n/(2*np.pi)
    )



    Mean_Of_Variances = (
        uncooled.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled[0].n/(2*np.pi)))**2,
        uncooled_1.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_1[0].n/(2*np.pi)))**2,
        uncooled_2.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_2[0].n/(2*np.pi)))**2,
        uncooled_3.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_3[0].n/(2*np.pi)))**2
    )
    Mean_Of_Variances_cooled = (
        cooled.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled[0].n/(2*np.pi)))**2,
        cooled_1.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_1[0].n/(2*np.pi)))**2,
        cooled_2.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_2[0].n/(2*np.pi)))**2,
        cooled_3.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[0],#/(omega_ref/(fit_uncooled_3[0].n/(2*np.pi)))**2
    )
    Var_Of_Variances =
(uncooled.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[1],uncooled_1.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[1],uncooled_2.calc_mean_and_variance_of_variances(NumberOfOscillations=10)
[1],uncooled_3.calc_mean_and_variance_of_variances(NumberOfOscillations=10)[1])

    del uncooled_1.voltage,uncooled_1.time
    del uncooled_2.voltage,uncooled_2.time
    del uncooled_3.voltage,uncooled_3.time
    del cooled_1.voltage, cooled_1.time
    del cooled_2.voltage, cooled_2.time
    del cooled_3.voltage, cooled_3.time

    pressures = (uncooled.pmbar, uncooled_1.pmbar, uncooled_2.pmbar, uncooled_3.pmbar)
    MultiPSD[1].set_title('PSDs_%imV'%(moglabs_voltages[index]))
    MultiPSD[0].savefig('PSDs/PSDs_%imV.pdf'%(moglabs_voltages[index]))

    plt.close('all')
    del  uncooled_1, uncooled_2, uncooled_3
    del  cooled_1, cooled_2, cooled_3
    #gc.collect()
    return (fit_uncooled, fit_uncooled_1, fit_uncooled_2, fit_uncooled_3, MultiPSD,
RawArea_uncooled, fit_cooled, fit_cooled_1, fit_cooled_2, fit_cooled_3, RawArea_cooled,
coolingfactor_calc_by_var, coolingfactor_calc_by_var_big_filter,uncooled, cooled,
Gamma_by_autocorrelation, Mean_Of_Variances, Var_Of_Variances, pressures,
Gamma_by_autocorrelation_cooled, Mean_Of_Variances_cooled,var_cooled_big_filter)

cpu_count = 8 #cpu_count() #8
ItemsPerThread = 1
workerPool = Pool(cpu_count)
data = workerPool.map(full_analysis, range(len(moglabs_voltages))) #,chunksize =
ItemsPerThread)
#data = full_analysis(1)
workerPool.close()
workerPool.terminate()
workerPool.join()
```

```
278
279 #data = [data]
280
281 for dataset_index in range(len(data)):
282     fit_uncooled = data[dataset_index][0]
283     fit_uncooled_1 = data[dataset_index][1]
284     fit_uncooled_2 = data[dataset_index][2]
285     fit_uncooled_3 = data[dataset_index][3]
286     MultiPSD = data[dataset_index][4]
287     RawArea_uncooled = data[dataset_index][5]
288     fit_cooled = data[dataset_index][6]
289     fit_cooled_1 = data[dataset_index][7]
290     fit_cooled_2 = data[dataset_index][8]
291     fit_cooled_3 = data[dataset_index][9]
292     RawArea_cooled = data[dataset_index][10]
293     coolingfactor_var.append(data[dataset_index][11])
294     coolingfactor_var_big_filter.append(data[dataset_index][12])
295     Gamma_by_autocorrelation.append(data[dataset_index][15])
296     Mean_Of_Variances.append(data[dataset_index][16])
297     Var_Of_Variances.append(data[dataset_index][17])
298     pressureList.append(data[dataset_index][18])
299     Gamma_by_autocorrelation_cooled.append(data[dataset_index][19])
300     Mean_Of_Variances_cooled.append(data[dataset_index][20])
301     Var_cooled_BigFilter.append(data[dataset_index][21])
302
uncooledFittedOmegas.append((fit_uncooled[0].n,fit_uncooled_1[0].n,fit_uncooled_2[0].n,fit_uncool

303
uncooledFittedOmegasErr.append((fit_uncooled[0].s,fit_uncooled_1[0].s,fit_uncooled_2[0].s,fit_unc

304     uncooledFittedOmegasFileset_0.append(fit_uncooled[0].n)
305     uncooledFittedOmegasFileset_1.append(fit_uncooled_1[0].n)
306     uncooledFittedOmegasFileset_2.append(fit_uncooled_2[0].n)
307     uncooledFittedOmegasFileset_3.append(fit_uncooled_3[0].n)
308
uncooledFittedAreas.append((fit_uncooled[1].n,fit_uncooled_1[1].n,fit_uncooled_2[1].n,fit_uncoole

309
uncooledFittedAreasErr.append((fit_uncooled[1].s,fit_uncooled_1[1].s,fit_uncooled_2[1].s,fit_unco

310
uncooledFittedGammas.append((fit_uncooled[2].n/2/np.pi,fit_uncooled_1[2].n/2/np.pi,fit_uncooled_2

311
uncooledFittedGammasUfloat.append((fit_uncooled[2]/2/np.pi,fit_uncooled_1[2]/2/np.pi,fit_uncooled

312
uncooledFits.append((fit_uncooled[3:5],fit_uncooled_1[3:5],fit_uncooled_2[3:5],fit_uncooled_3[3:5

313
cooledFittedOmegas.append((fit_cooled[0].n,fit_cooled_1[0].n,fit_cooled_2[0].n,fit_cooled_3[0].n)

314
cooledFittedOmegasErr.append((fit_cooled[0].s,fit_cooled_1[0].s,fit_cooled_2[0].s,fit_cooled_3[0]

315     cooledFittedOmegasFileset_0.append(fit_cooled[0].n)
316     cooledFittedOmegasFileset_1.append(fit_cooled_1[0].n)
317
cooledFittedAreas.append((fit_cooled[1].n,fit_cooled_1[1].n,fit_cooled_2[1].n,fit_cooled_3[1].n))

318
cooledFittedAreasErr.append((fit_cooled[1].s,fit_cooled_1[1].s,fit_cooled_2[1].s,fit_cooled_3[1].

319
cooledFittedGammas.append((fit_cooled[2].n/2/np.pi,fit_cooled_1[2].n/2/np.pi,fit_cooled_2[2].n/2/

320
cooledFittedGammasUfloat.append((fit_cooled[2]/2/np.pi,fit_cooled_1[2]/2/np.pi,fit_cooled_2[2]/2/

321
cooledFits.append((fit_cooled[3:5],fit_cooled_1[3:5],fit_cooled_2[3:5],fit_cooled_3[3:5]))
```

```
    uncooledRawAreas.append(RawArea_uncooled)
    cooledRawAreas.append(RawArea_cooled)
    PSDs.append(MultiPSD)

coolingfactor_gamma = (np.array(cooledFittedGammas)/np.array(uncooledFittedGammas))
coolingfactor_gamma_ufloat =
(np.array(cooledFittedGammasUfloat)/np.array(uncooledFittedGammasUfloat))
coolingfactor = (np.array(uncooledRawAreas)/np.array(cooledRawAreas))
coolingfactor_GammaAuto =
(np.array(Gamma_by_autocorrelation_cooled)/np.array(Gamma_by_autocorrelation))
GammaAutoDiff = (np.array(Gamma_by_autocorrelation_cooled) -
np.array(Gamma_by_autocorrelation))
GammaSpecDiff = (np.array(cooledFittedGammas) - np.array(uncooledFittedGammas))

coolingfactor_gamma_err = []
for i in range(len(coolingfactor_gamma_ufloat)):

coolingfactor_gamma_err.append((coolingfactor_gamma_ufloat[i,0].s,coolingfactor_gamma_ufloat[i,1]

np.save(saving_dir+'/coolingfactorGamma',coolingfactor_gamma)
np.save(saving_dir+'/coolingfactorGammaErr',coolingfactor_gamma_err)
np.save(saving_dir+'/VarCooledBigFilter',np.array(Var_cooled_BigFilter))

np.save(saving_dir+'/VarUncooledBigFilter',np.array(Var_cooled_BigFilter)*np.array(coolingfactor_

aList=[]
for i in range(len(uncooledFittedOmegas[0])):
    aList.append(np.array(uncooledFittedOmegas)[:,i]/np.max(np.array(uncooledFittedOmegas)
[:,i]))

aArray = np.transpose(np.array(aList))
aTheoryArray =
np.linspace(min(np.ndarray.flatten(aArray))/max(np.ndarray.flatten(aArray)),1,100)

np.savetxt('uncooledFittedOmegasFileset_0.dat',(uncooledFittedOmegasFileset_0),
header='fitted Omegas')
np.savetxt('uncooledFittedOmegasFileset_1.dat',(uncooledFittedOmegasFileset_1),
header='fitted Omegas')
np.savetxt('uncooledFittedOmegasFileset_2.dat',(uncooledFittedOmegasFileset_2),
header='fitted Omegas')
np.savetxt('uncooledFittedOmegasFileset_3.dat',(uncooledFittedOmegasFileset_3),
header='fitted Omegas')

np.savetxt('coolingfactor.dat', ([x[0] for x in coolingfactor],[x[1] for x in
coolingfactor],[x[2] for x in coolingfactor],[x[3] for x in coolingfactor],[x[0] for x in
coolingfactor_gamma],[x[1] for x in coolingfactor_gamma],[x[2] for x in coolingfactor_gamma],
[x[3] for x in coolingfactor_gamma],[x[0] for x in coolingfactor_var],[x[1] for x in
coolingfactor_var],[x[2] for x in coolingfactor_var],[x[3] for x in coolingfactor_var],[x[0] for
x in coolingfactor_var_big_filter],[x[1] for x in coolingfactor_var_big_filter],[x[2] for x in
coolingfactor_var_big_filter],[x[3] for x in coolingfactor_var_big_filter]),
header='RawAreaCoolingFactors dataset_0 \t dataset_1 \t dataset_2 \t dataset_3 \t
GammaCoolingFactors dataset_0 \t dataset_1 \t dataset_2 \t dataset_3 \t VarCoolingFactors
dataset_0 \t dataset_1 \t dataset_2 \t dataset_3 \t BigFilterVarCoolingFactors dataset_0 \t
dataset_1 \t dataset_2 \t dataset_')

FittedAreaVSPower_fig = plt.figure()
FittedAreaVSPower_ax = FittedAreaVSPower_fig.add_subplot(111)
FittedAreaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedAreas)
[:,0],'o',label='dataset 0')
FittedAreaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedAreas)
[:,1],'o',label='dataset 1')
FittedAreaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedAreas)
[:,2],'o',label='dataset 2')
FittedAreaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedAreas)
[:,3],'o',label='dataset 3')

FittedAreaVSPower_ax.set_xlabel('Power (mW)')
FittedAreaVSPower_ax.set_ylabel('Fitted Area')
FittedAreaVSPower_ax.legend(loc='best')
```

```
366 FittedAreaVSPower = FittedAreaVSPower_fig, FittedAreaVSPower_ax
367 np.save(saving_dir+'/FittedAreaUncooled',np.array(uncooledFittedAreas))
368 np.save(saving_dir+'/FittedAreaUncooledErr',np.array(uncooledFittedAreasErr))
369 np.save(saving_dir+'/FittedAreaCooled',np.array(cooledFittedAreas))
370 np.save(saving_dir+'/FittedAreaCooledErr',np.array(cooledFittedAreasErr))
371
372 FittedGammaVSPower_fig = plt.figure()
373 FittedGammaVSPower_ax = FittedGammaVSPower_fig.add_subplot(111)
374 FittedGammaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedGammas)
[:,0],'o',label='dataset 0')
375 FittedGammaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedGammas)
[:,1],'o',label='dataset 1')
376 FittedGammaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedGammas)
[:,2],'o',label='dataset 2')
377 FittedGammaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedGammas)
[:,3],'o',label='dataset 3')
378
379 FittedGammaVSPower_ax.set_xlabel('Power (mW)')
380 FittedGammaVSPower_ax.set_ylabel('Fitted Gamma (Hz)')
381 FittedGammaVSPower_ax.legend(loc='best')
382 FittedGammaVSPower = FittedGammaVSPower_fig, FittedGammaVSPower_ax
383
384 RawAreaVSPower_fig = plt.figure()
385 RawAreaVSPower_ax = RawAreaVSPower_fig.add_subplot(111)
386 RawAreaVSPower_ax.plot(trapping_powers,uncooledRawAreas,'o')
387 RawAreaVSPower_ax.set_xlabel('Power (mW)')
388 RawAreaVSPower_ax.set_ylabel(r'Area')
389 RawAreaVSPower = RawAreaVSPower_fig, RawAreaVSPower_ax
390 np.save(saving_dir+'/RawAreaUncooled',np.array(uncooledRawAreas))
391
392 def omega_power_fit(TrappingPowers,constant):
393     return constant*np.sqrt(TrappingPowers)
394
395 FittedOmegaVSPower_fig = plt.figure()
396 FittedOmegaVSPower_ax = FittedOmegaVSPower_fig.add_subplot(111)
397 FittedOmegaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedOmegas)
[:,0]/(2*np.pi),'o',label='dataset 0',color='blue')
398 uncooled_omega_power_fit , _ = optoanalysis.fit_curvefit([70],trapping_powers,
np.array(uncooledFittedOmegas)[:,0]/(2*np.pi), omega_power_fit)
399
#FittedOmegaVSPower_ax.plot(trapping_powers,omega_power_fit(trapping_powers,uncooled_omega_power_
',color='blue')
400 FittedOmegaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedOmegas)
[:,1]/(2*np.pi),'o',label='dataset 1',color='orange')
401 uncooled_1_omega_power_fit , _ = optoanalysis.fit_curvefit([70],trapping_powers,
np.array(uncooledFittedOmegas)[:,1]/(2*np.pi), omega_power_fit)
402
#FittedOmegaVSPower_ax.plot(trapping_powers,omega_power_fit(trapping_powers,uncooled_1_omega_powe
',color='orange')
403 FittedOmegaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedOmegas)
[:,2]/(2*np.pi),'o',label='dataset 2',color='green')
404 uncooled_2_omega_power_fit , _ = optoanalysis.fit_curvefit([70],trapping_powers,
np.array(uncooledFittedOmegas)[:,2]/(2*np.pi), omega_power_fit)
405
#FittedOmegaVSPower_ax.plot(trapping_powers,omega_power_fit(trapping_powers,uncooled_2_omega_powe
',color='green')
406 FittedOmegaVSPower_ax.plot(trapping_powers,np.array(uncooledFittedOmegas)
[:,3]/(2*np.pi),'o',label='dataset 3',color='red')
407 uncooled_3_omega_power_fit , _ = optoanalysis.fit_curvefit([70],trapping_powers,
np.array(uncooledFittedOmegas)[:,3]/(2*np.pi), omega_power_fit)
408
#FittedOmegaVSPower_ax.plot(trapping_powers,omega_power_fit(trapping_powers,uncooled_3_omega_powe
',color='red')
409 FittedOmegaVSPower_ax.set_xlabel('Power (mW)')
410 FittedOmegaVSPower_ax.set_ylabel('Frequency (Hz)')
411 FittedOmegaVSPower_ax.legend(loc='best')
412 FittedOmegaVSPower = FittedOmegaVSPower_fig, FittedOmegaVSPower_ax
413
414 np.save(saving_dir+'/uncooledFittedOmegas',np.array(uncooledFittedOmegas)/2/np.pi)
415 np.save(saving_dir+'/uncooledFittedOmegasErr',np.array(uncooledFittedOmegasErr))
416 np.save(saving_dir+'/cooledFittedOmegas',np.array(cooledFittedOmegas)/2/np.pi)
```

```
417 np.save(saving_dir+'/cooledFittedOmegasErr',np.array(cooledFittedOmegasErr))
418
419 GammaAutoVSPower_fig = plt.figure()
420 GammaAutoVSPower_ax = GammaAutoVSPower_fig.add_subplot(111)
421 GammaAutoVSPower_ax.plot(trapping_powers,Gamma_by_autocorrelation,'o')
422 GammaAutoVSPower_ax.set_xlabel('Power (mW)')
423 GammaAutoVSPower_ax.set_ylabel('Gamma (Hz)')
424 GammaAutoVSPower = GammaAutoVSPower_fig, GammaAutoVSPower_ax
425
426 MeanOfVarVSPower_fig = plt.figure()
427 MeanOfVarVSPower_ax = MeanOfVarVSPower_fig.add_subplot(111)
428 MeanOfVarVSPower_ax.plot(trapping_powers,Mean_Of_Variances,'o')
429 MeanOfVarVSPower_ax.set_xlabel('Power (mW)')
430 MeanOfVarVSPower_ax.set_ylabel(r'Mean of Variances $(V^2)$')
431 MeanOfVarVSPower = MeanOfVarVSPower_fig, MeanOfVarVSPower_ax
432 np.save(saving_dir+'/MeanOfVar',np.array(Mean_Of_Variances))
433
434 VarOfVarVSPower_fig = plt.figure()
435 VarOfVarVSPower_ax = VarOfVarVSPower_fig.add_subplot(111)
436 VarOfVarVSPower_ax.plot(trapping_powers,Var_Of_Variances,'o')
437 VarOfVarVSPower_ax.set_xlabel('Power (mW)')
438 VarOfVarVSPower_ax.set_ylabel(r'Variance of Variances ($V^4$)')
439 VarOfVarVSPower = VarOfVarVSPower_fig, VarOfVarVSPower_ax
440
441 PressureVSPower_fig = plt.figure()
442 PressureVSPower_ax = PressureVSPower_fig.add_subplot(111)
443 PressureVSPower_ax.plot(trapping_powers,pressureList,'o')
444 PressureVSPower_ax.set_xlabel('Power (mW)')
445 PressureVSPower_ax.set_ylabel('Pressure (mbar)')
446 PressureVSPower = PressureVSPower_fig, PressureVSPower_ax
447
448 GammaAutoVSPressure_fig = plt.figure()
449 GammaAutoVSPressure_ax = GammaAutoVSPressure_fig.add_subplot(111)
450 GammaAutoVSPressure_ax.plot(pressureList,Gamma_by_autocorrelation,'o')
451 GammaAutoVSPressure_ax.set_xlabel('Pressure (mbar)')
452 GammaAutoVSPressure_ax.set_ylabel('Gamma (Hz)')
453 GammaAutoVSPressure = GammaAutoVSPressure_fig, GammaAutoVSPressure_ax
454
455 RawCoolingVSPower_fig = plt.figure()
456 RawCoolingVSPower_ax = RawCoolingVSPower_fig.add_subplot(111)
457 RawCoolingVSPower_ax.plot(trapping_powers,coolingfactor,'o')
458 RawCoolingVSPower_ax.set_xlabel('Power (mW)')
459 RawCoolingVSPower_ax.set_ylabel('Cooling Factor')
460 RawCoolingVSPower = RawCoolingVSPower_fig, RawCoolingVSPower_ax
461
462 def CoolingVSaArrayFit(aArray,Gain,N):
463     return  1 + (-Gain)*np.sin((np.pi+np.pi/2+2*np.pi*int(N))*aArray)
464
465 GammaCoolingVSPower_fig = plt.figure()
466 GammaCoolingVSPower_ax = GammaCoolingVSPower_fig.add_subplot(111)
467 GammaCoolingVSPower_ax.plot(aArray,coolingfactor_gamma,'o')
468 cooling_a_fit , cooling_a_fit_cov = optoanalysis.fit_curvefit([3,1], aArray[:,0],
np.array(coolingfactor_gamma[:,0]), CoolingVSaArrayFit)
469
GammaCoolingVSPower_ax.plot(aTheoryArray,CoolingVSaArrayFit(aTheoryArray,cooling_a_fit[0],cooling
', label=r'dataset 0: 1+(-%.4f)$\cdot \sin((\pi/2+3 \pi$ %i)$\cdot \lambda$)'%
(cooling_a_fit[0],cooling_a_fit[1]), color='blue')
470 np.save(saving_dir+'/cooling_a_fit_cov',cooling_a_fit_cov)
471 cooling_a_fit_1 , _ = optoanalysis.fit_curvefit([3,1], aArray[:,1],
np.array(coolingfactor_gamma[:,1]), CoolingVSaArrayFit)
472
GammaCoolingVSPower_ax.plot(aTheoryArray,CoolingVSaArrayFit(aTheoryArray,cooling_a_fit_1[0],cooli
', label=r'dataset 1: 1+(-%.4f)$\cdot \sin((\pi/2+3 \pi$ %i)$\cdot \lambda$)'%
(cooling_a_fit_1[0],cooling_a_fit_1[1]), color='orange')
473 cooling_a_fit_2 , _ = optoanalysis.fit_curvefit([3,1], aArray[:,2],
np.array(coolingfactor_gamma[:,2]), CoolingVSaArrayFit)
474
GammaCoolingVSPower_ax.plot(aTheoryArray,CoolingVSaArrayFit(aTheoryArray,cooling_a_fit_2[0],cooli
', label=r'dataset 2: 1+(-%.4f)$\cdot \sin((\pi/2+3 \pi$ %i)$\cdot \lambda$)'%
(cooling_a_fit_2[0],cooling_a_fit_2[1]), color='green')
475 cooling_a_fit_3 , _ = optoanalysis.fit_curvefit([3,1], aArray[:,3],
```

```
np.array(coolingfactor_gamma[:,3]), CoolingVSaArrayFit)

GammaCoolingVSPower_ax.plot(aTheoryArray,CoolingVSaArrayFit(aTheoryArray,cooling_a_fit_3[0],cooli
', label=r'dataset 3: 1+(-%.4f)$\cdot \sin((\pi/2+3 \pi$ %i)$\cdot \lambda$)'%
(cooling_a_fit_3[0],cooling_a_fit_3[1]), color='red')
GammaCoolingVSPower_ax.set_xlabel(r'$\omega / \omega_0 = \lambda$')
GammaCoolingVSPower_ax.set_ylabel('Cooling Factor')
GammaCoolingVSPower_ax.legend(loc='best')
GammaCoolingVSPower = GammaCoolingVSPower_fig, GammaCoolingVSPower_ax

VarCoolingVSPower_fig = plt.figure()
VarCoolingVSPower_ax = VarCoolingVSPower_fig.add_subplot(111)
VarCoolingVSPower_ax.plot(trapping_powers,coolingfactor_var,'o')
VarCoolingVSPower_ax.set_xlabel('Power (mW)')
VarCoolingVSPower_ax.set_ylabel('Cooling Factor')
VarCoolingVSPower = VarCoolingVSPower_fig, VarCoolingVSPower_ax

BigFilterVarCoolingVSPower_fig = plt.figure()
BigFilterVarCoolingVSPower_ax = BigFilterVarCoolingVSPower_fig.add_subplot(111)
BigFilterVarCoolingVSPower_ax.plot(aArray,coolingfactor_var_big_filter,'o')
BigFilterVarCoolingVSPower_ax.set_xlabel(r'$\omega / \omega_0$')
BigFilterVarCoolingVSPower_ax.set_ylabel('Cooling Factor')
BigFilterVarCoolingVSPower = BigFilterVarCoolingVSPower_fig, BigFilterVarCoolingVSPower_ax

np.save(saving_dir+'/BigFilterVarCoolingFactors',coolingfactor_var_big_filter)

GammaAutoCoolingVSPower_fig = plt.figure()
GammaAutoCoolingVSPower_ax = GammaAutoCoolingVSPower_fig.add_subplot(111)
GammaAutoCoolingVSPower_ax.plot(aArray,coolingfactor_GammaAuto,'o')
GammaAutoCoolingVSPower_ax.set_xlabel(r'$\omega / \omega_0$')
GammaAutoCoolingVSPower_ax.set_ylabel('Cooling Factor')
GammaAutoCoolingVSPower = GammaAutoCoolingVSPower_fig, GammaAutoCoolingVSPower_ax

GammaEff_Gamma0VSPower_fig = plt.figure()
GammaEff_Gamma0VSPower_ax = GammaEff_Gamma0VSPower_fig.add_subplot(111)
GammaEff_Gamma0VSPower_ax.plot(aArray,GammaAutoDiff,'o')

def GammaEff_Gamma0VSPowerFit(aArray,GammaMax, N):
    return GammaMax * np.sin((np.pi/2+3*np.pi*int(N))*aArray)
uncooled_gamma_fb_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,0],
np.array(GammaAutoDiff[:,0]), GammaEff_Gamma0VSPowerFit)

GammaEff_Gamma0VSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_gamma
', label=r'dataset 0: -%i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_gamma_fb_power_fit[0],uncooled_gamma_fb_power_fit[1]), color='blue')
uncooled_1_gamma_fb_power_fit , _ = optoanalysis.fit_curvefit([6000,1],aArray[:,1],
np.array(GammaAutoDiff[:,1]), GammaEff_Gamma0VSPowerFit)

GammaEff_Gamma0VSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_1_gam
', label=r'dataset 1: -%i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_1_gamma_fb_power_fit[0],uncooled_1_gamma_fb_power_fit[1]),color='orange')
uncooled_2_gamma_fb_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,2],
np.array(GammaAutoDiff[:,2]), GammaEff_Gamma0VSPowerFit)

GammaEff_Gamma0VSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_2_gam
',label=r'dataset 2: -%i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_2_gamma_fb_power_fit[0],uncooled_2_gamma_fb_power_fit[1]),color='green')
uncooled_3_gamma_fb_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,3],
np.array(GammaAutoDiff[:,3]), GammaEff_Gamma0VSPowerFit)

GammaEff_Gamma0VSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_3_gam
',label=r'dataset 3: -%i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_3_gamma_fb_power_fit[0],uncooled_3_gamma_fb_power_fit[1]),color='red')
GammaEff_Gamma0VSPower_ax.set_xlabel(r'$\omega / \omega_0$')
GammaEff_Gamma0VSPower_ax.set_ylabel(r'$\Gamma_{fb}=\Gamma_{Eff}-\Gamma_0$')
GammaEff_Gamma0VSPower_ax.legend(loc='best')
GammaEff_Gamma0VSPower = GammaEff_Gamma0VSPower_fig,  GammaEff_Gamma0VSPower_ax

GammaEff_Gamma0_SpecVSPower_fig = plt.figure()
GammaEff_Gamma0_SpecVSPower_ax = GammaEff_Gamma0_SpecVSPower_fig.add_subplot(111)
GammaEff_Gamma0_SpecVSPower_ax.plot(aArray,GammaSpecDiff,'o')
```

```
527 uncooled_gamma_fb_spec_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,0],
np.array(GammaSpecDiff[:,0]), GammaEff_Gamma0VSPowerFit)
528
GammaEff_Gamma0_SpecVSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_
', label=r'dataset 0: %i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_gamma_fb_spec_power_fit[0],uncooled_gamma_fb_spec_power_fit[1]), color='blue')
529 uncooled_1_gamma_fb_spec_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,1],
np.array(GammaSpecDiff[:,1]), GammaEff_Gamma0VSPowerFit)
530
GammaEff_Gamma0_SpecVSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_
', label=r'dataset 1: %i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_1_gamma_fb_spec_power_fit[0],uncooled_1_gamma_fb_spec_power_fit[1]),color='orange')
531 uncooled_2_gamma_fb_spec_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,2],
np.array(GammaSpecDiff[:,2]), GammaEff_Gamma0VSPowerFit)
532
GammaEff_Gamma0_SpecVSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_
',label=r'dataset 2: %i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_2_gamma_fb_spec_power_fit[0],uncooled_2_gamma_fb_spec_power_fit[1]),color='green')
533 uncooled_3_gamma_fb_spec_power_fit , _ = optoanalysis.fit_curvefit([6000,1], aArray[:,3],
np.array(GammaSpecDiff[:,3]), GammaEff_Gamma0VSPowerFit)
534
GammaEff_Gamma0_SpecVSPower_ax.plot(aTheoryArray,GammaEff_Gamma0VSPowerFit(aTheoryArray,uncooled_
',label=r'dataset 3: %i$\cdot \sin(\pi/2+3 \pi$ %i)'%
(uncooled_3_gamma_fb_spec_power_fit[0],uncooled_3_gamma_fb_spec_power_fit[1]),color='red')
535 GammaEff_Gamma0_SpecVSPower_ax.set_xlabel(r'$\omega / \omega_0$')
536 GammaEff_Gamma0_SpecVSPower_ax.set_ylabel(r'$\gamma_\mathrm{fb}=\gamma_\mathrm{eff}-
\gamma_0$ (Hz)')
537 GammaEff_Gamma0_SpecVSPower_ax.legend(loc='best')
538 GammaEff_Gamma0_SpecVSPower = GammaEff_Gamma0_SpecVSPower_fig,
GammaEff_Gamma0_SpecVSPower_ax
539
540 GammaOmegaSquaredScaledAreaVSPower_fig = plt.figure()
541 GammaOmegaSquaredScaledAreaVSPower_ax =
GammaOmegaSquaredScaledAreaVSPower_fig.add_subplot(111)
542 omega_ref=476052
543 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(uncooledFittedOmegas)
[:,0]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation)[:,0]*np.array(uncooledRawAreas)
[:,0],'o',label='0', color='blue')
544 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,0]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation_cooled)[:,0]*np.array(cooledRawAreas)
[:,0],'o',label='0', markerfacecolor='none', markeredgecolor='blue')
545 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(uncooledFittedOmegas)
[:,1]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation)[:,1]*np.array(uncooledRawAreas)
[:,1],'o',label='1', color='orange')
546 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,1]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation_cooled)[:,1]*np.array(cooledRawAreas)
[:,1],'o',label='1', markerfacecolor='none', markeredgecolor='orange')
547 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(uncooledFittedOmegas)
[:,2]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation)[:,2]*np.array(uncooledRawAreas)
[:,2],'o',label='2', color='green')
548 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,2]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation_cooled)[:,2]*np.array(cooledRawAreas)
[:,2],'o',label='2', markerfacecolor='none', markeredgecolor='green')
549 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(uncooledFittedOmegas)
[:,3]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation)[:,3]*np.array(uncooledRawAreas)
[:,3],'o',label='3', color='red')
550 GammaOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,3]/(2*np.pi))**2*np.array(Gamma_by_autocorrelation_cooled)[:,3]*np.array(cooledRawAreas)
[:,3],'o',label='3', markerfacecolor='none', markeredgecolor='red')
551 GammaOmegaSquaredScaledAreaVSPower_ax.set_xlabel('Power (mW)')
552 GammaOmegaSquaredScaledAreaVSPower_ax.set_ylabel(r'$\Gamma \omega^2 A_{raw}$')
553 GammaOmegaSquaredScaledAreaVSPower = GammaOmegaSquaredScaledAreaVSPower_fig,
GammaOmegaSquaredScaledAreaVSPower_ax
554
555 GammaSpecOmegaSquaredScaledAreaVSPower_fig = plt.figure()
556 GammaSpecOmegaSquaredScaledAreaVSPower_ax =
GammaSpecOmegaSquaredScaledAreaVSPower_fig.add_subplot(111)
557 GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,
(np.array(uncooledFittedOmegas)[:,0]/(2*np.pi))**2*np.array(uncooledFittedGammas)
[:,0]*np.array(uncooledRawAreas)[:,0],'o',label='0', color='blue')
558 GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
```

```
[:,0]/(2*np.pi))**2*np.array(cooledFittedGammas)[:,0]*np.array(cooledRawAreas)
[:,0],'o',label='0', markerfacecolor='none', markeredgecolor='blue')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,
(np.array(uncooledFittedOmegas)[:,1]/(2*np.pi))**2*np.array(uncooledFittedGammas)
[:,1]*np.array(uncooledRawAreas)[:,1],'o',label='1', color='orange')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,1]/(2*np.pi))**2*np.array(cooledFittedGammas)[:,1]*np.array(cooledRawAreas)
[:,1],'o',label='1', markerfacecolor='none', markeredgecolor='orange')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,
(np.array(uncooledFittedOmegas)[:,2]/(2*np.pi))**2*np.array(uncooledFittedGammas)
[:,2]*np.array(uncooledRawAreas)[:,2],'o',label='2', color='green')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,2]/(2*np.pi))**2*np.array(cooledFittedGammas)[:,2]*np.array(cooledRawAreas)
[:,2],'o',label='2', markerfacecolor='none', markeredgecolor='green')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,
(np.array(uncooledFittedOmegas)[:,3]/(2*np.pi))**2*np.array(uncooledFittedGammas)
[:,3]*np.array(uncooledRawAreas)[:,3],'o',label='3', color='red')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.plot(trapping_powers,(np.array(cooledFittedOmegas)
[:,3]/(2*np.pi))**2*np.array(cooledFittedGammas)[:,3]*np.array(cooledRawAreas)
[:,3],'o',label='3', markerfacecolor='none', markeredgecolor='red')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.set_xlabel('Power (mW)')
GammaSpecOmegaSquaredScaledAreaVSPower_ax.set_ylabel(r'$\gamma_\mathrm{eff}
\omega_\mathrm{eff}^2 A_\mathrm{Raw}$')
GammaSpecOmegaSquaredScaledAreaVSPower = GammaSpecOmegaSquaredScaledAreaVSPower_fig,
GammaSpecOmegaSquaredScaledAreaVSPower_ax

RoomTemp =
optoanalysis.arrange_plots_on_one_canvas([FittedAreaVSPower,FittedGammaVSPower,RawAreaVSPower,Fit
 GammaAutoVSPressure],SubtitleArray=['Fitted Area VS Power', 'Fitted Gamma VS Power', 'Area
under raw PSD VS Power','Fitted Frequency VS Power','Gamma using autocorrelation VS Power','Mean
of Varainces VS Power','Varaince of Variances VS Power', 'Gamma using autocorrelation VS
Pressure'],show_fig=False)
RoomTemp.savefig('RoomTempPowerChange.pdf')

CoolingEva =
optoanalysis.arrange_plots_on_one_canvas([RawCoolingVSPower,GammaCoolingVSPower,VarCoolingVSPower
 PressureVSPower],SubtitleArray=['Cooling Factor from Raw Area VS Power', 'Cooling Factor from
fitted Gamma VS Power', 'Cooling Factor from Variance of filtered data \n (PeakWidth=40kHz)',
'Cooling Factor from Variance of filtered data \n (center=365kHz, PeakWidth=90kHz)','Cooling
Factor from Gamma using autocorrelation', 'Pressure VS Power'],show_fig=False)
CoolingEva.savefig('CoolingEva.pdf',bbox_inches='tight', pad_inches=10)

Crosscheck =
optoanalysis.arrange_plots_on_one_canvas([GammaEff_Gamma0VSPower,GammaOmegaSquaredScaledAreaVSPow
['Gamma_Eff - Gamma_0','Gamma_Eff * Omega^2 * RawArea'],show_fig=False)
Crosscheck.savefig('Crosscheck.png')

CrosscheckSpec =
optoanalysis.arrange_plots_on_one_canvas([GammaEff_Gamma0_SpecVSPower,GammaSpecOmegaSquaredScaled
['Gamma_Eff - Gamma_0 using Spectrum','Gamma_Eff_Spec * Omega^2 * RawArea'],show_fig=False)
CrosscheckSpec.savefig('CrosscheckSpec.eps',bbox_inches='tight',pad_inches=10)

rescaledPSDs = optoanalysis.multi_plot_PSD([data[0][13],data[0][14],data[4][13],data[4]
[14],data[9][13],data[9][14],data[14][13],data[14][14],data[19][13],data[19][14]],LabelArray=
['860mW','860mW with FB','841mW','841mW with FB','822mW','833mW with FB','803mW','803mW with
FB','784mW','784mW with FB'],alphaArray=[.3,.3,.3,.3,.3,.3,.3,.3,.3,.3],show_fig=False)
rescaledPSDs[1].set_title(r'after normalisation with the monitor output')
rescaledPSDs[0].savefig('rescaledPSDs.pdf')

QVSPower_fig = plt.figure()
QVSPower_ax = QVSPower_fig.add_subplot(111)

QVSPower_ax.plot(aArray,np.array(uncooledFittedOmegas)/np.array(uncooledFittedGammas)/2/np.pi,'o'

QVSPower_ax.set_xlabel('Power (mW)')
QVSPower_ax.set_ylabel('Quality Factor')
QVSPower = QVSPower_fig, QVSPower_ax
QVSPower_fig.savefig('QualityFactors.pdf')

```

C.2. C++ code

C.2.1. Code commentary

The C++ code printed in the following subsection was used to calculate the normalised free energy difference by using the three different evaluation methods depicted in figure 20 and 21. One of these three methods was implemented in the from of the fluctuation theorem and can be seen in code line 203. For the entire evaluation the script must not only be provided with the trajectories where the thermodynamic protocols are taking place, but also the mechanical model and the inverse heat bath temperature model over the course of one single trajectory has to be entered as it is demanded in lines 43-46, 139 and 149-151 of the printed code.
The stronghold of the C++ program is that is was making it possible to evaluate an entire set of thermodynamic protocols, which means processing 15000 trajectories for each of the 9 different protocol times, so 135000 trajectories in total, in a matter of seconds. This was possible due to the improvement in computation time efficiency marked by the implementation of a multiplatform shared memory multiprocessing application programming interface called OpenMP [65, 66]. Thus making it easy to distribute the computations to several processing units, also known as computer cores, and as a consequence drastically reduce the computation time.
The implementation of the OpenMP interface goes as follows. First we have to create a function which in this case is called `Jarzynski` and does all the necessary processing for one protocol time, as can be seen starting in code line 42. Thus this subscript evaluates one block of 15000 trajectories. Similarly to the Python multiprocessing unit, we again have to make sure that we only have one variable in the return statement of the called function, as it is written in code line 214. For this specific purpose we defined a custom type encapsulating all calculated results, which is printed in code line 18, in order to masquerade all of the different variables as one single variable, which is returned to the main program. The central script then has the OpenMP function in code line 225, which calls the before introduced subfunction called `Jarzynski` in a parallel manner and manages the workload distribution to the different cores and the RAM space assignment to the individual threads. Finally after the OpenMP function has done its job, the C++ script saves all the calculated values in a text file.

Code

```
#include <vector>
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <omp.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <time.h>
#include <string>
#include <sstream>
#include <fstream>
#include <ctime>
#include <stdio.h>
#include <time.h>
#include <sstream>
#include <random>

typedef struct thermodynamics {
        double inverseTau;
        double expectedFreeEnergy;
        double freeEnergy;
        double meanWork;
        double forLinearResponse;
}thermo_type;

using namespace std;

void printMatrix(std::vector<std::vector<double>>matrix){
  int i,j;
  int N = matrix.size();
  int NL = 2; //matrix[0].size();
  for(i=0; i<N;i++){
    std::cout << "Line " << i << " :\t";
    for(j=0; j<NL; j++){
      printf("%.15e,\t",matrix[i][j]);
    }
    printf("\n");
  }
  printf("\n");
}

thermo_type Jarzynski( int l) {
  double a_0 = 0.90;        //starting potential
  double a_1 = 1;        //ending potential
  double beta_0 = 1- 2.5545 * sin ((7*M_PI/2)*a_0);      //starting beta
  double beta_1 = 1- 2.5545 * sin ((7*M_PI/2)*a_1);      //ending beta
  double k = 2;
  double m = 10;
  std::vector<int> ROI { 226,22624,461,297,903,256,353,610,1738 };
  std::vector<int> run {1,2,3,4,5,6,7,8,9};

  std::vector<double> PositionNorm {5.00137, 5.00137, 5.00137, 5.00137, 5.00137,
5.00137, 5.00137, 5.00137, 5.00137};
  std::vector<double> VelocityNorm {9.50278, 9.50278, 9.50278, 9.50278, 9.50278,
9.50278, 9.50278, 9.50278, 9.50278};


  int protocolLength = ROI[l];   // length of the time protocol, which will be
varied later, comparable to the ROI parameter

  std::string Line;
  int ColNo;
  std::string value;
  std::vector <std::vector <double>> data;
```

```
  int LineNo = 0;

  for (int i=1; i<=4; i++){
    ostringstream os;
    os << "/media/fastRaid0/people/markus/20180625 - WSE/short-filtered/WSE_run" <<
(l+1) << " - tt - short_" << i << ".dat";
    std::string Filename = os.str();
    //std::cout << Filename << "\n";
    if((l==1)&&(i==7 || i==10 || i==15)){
      std::cout << "SKIPPED" << std::endl;
      continue;}
    std::ifstream InputFile(Filename);
    while(std::getline(InputFile, Line)){
      std::istringstream LineStream(Line);
      //ColNo = 0;
      std::vector<double> tmpvec;
      while (std::getline(LineStream, value, '\t')){
        //std::cout << "value: " << value << "\n";
        if (value.empty() != 1){
          //if (i==1 && LineNo==0){std::cout << value << "\n";}
          //if (i==1 && LineNo==0){std::cout << stof(value) << "\n";}
          //std::cout << value << "\n";
          //std::cout << stof(value) << "\n";
          tmpvec.push_back(stof(value));
          //if (i==1 && LineNo==4){printMatrix(data);}
          //ColNo++;
        }
      }
      /*for( auto i: tmpvec){
        std::cout << i << std::endl;
        }
        std::cout << "//" << std::endl;*/
      data.push_back(tmpvec);
      //if(LineNo == 10){printMatrix(data);}
      LineNo++;
      //std::cout << LineNo << std::endl;
    }
  }

  //printMatrix(data);
  //std::cout << "data: " << data[3][0] << std::endl;

  //std::cout << "sizeof(data): " << data.size() << std::endl;
  //std::cout << ROI[l] << std::endl;
  std::cout << "RUN " << l+1 << " LOADED: " << data.size()/ROI[l] << "
trajectories" << std::endl;

  std::vector<double> velocity(data.size());

  velocity[0] = ((data[1][0])-(data[0][0]));
  for (int i = 1 ; i < data.size(); i++){
    velocity[i]=(((data[i][0])-(data[i-1][0])));
    //std::cout << velocity[i-1] << std::endl;
    //printf("%3e \n", velocity[0]);
    //printf("%3e \n", (data[i][1])-(data[i-1][1]));
  }

  int NumberOfPaths = 15000;
  double protocolTime = 100.0;
  thermo_type th;
  vector<double> initialDistr(NumberOfPaths);
  vector<double> initialDistr_v(NumberOfPaths);
  vector<double> EndDistr(NumberOfPaths);
  vector<double> EndDistr_v(NumberOfPaths);
  vector<double> energyonetraj(protocolLength);
```

```
125   vector<double> energy_slow(protocolLength);
126   vector<double> energy_fast(protocolLength);
127
128   double meanWork = 0;
129   double freeEnergy = 0;
130   double forLinearResponse = 0;
131
132   for (int j = 1; j <= NumberOfPaths; j++) {
133     if (l==1 && j % 1000 == 0) {cout << "Path number " << j << std::endl;} // for
seeing the progress in the command-line
134     double work = 0;
135     double z_1;
136     double z_2;
137
138     for (int i = 1; i <= protocolLength; i++) { // now we introduce the system to
thermal changes in this part
139       double a = (a_0 + (a_1 - a_0)*i / protocolLength);
140       z_1 = ((data[((j-1)*ROI[l]+(i-1))][0])*PositionNorm[l]);
141       z_2 = (velocity[((j-1)*ROI[l]+(i-1))])*VelocityNorm[l];
142       //std::cout << z_2 << std::endl;
143       //printf("%3e \n", z_2);
144       if (i==1) {
145         initialDistr[j]=(z_1);
146         initialDistr_v[j]=(z_2);
147       }
148
149       double beta = 1- 2.5545 * sin ((7*M_PI/2)*a); // (beta_0 + (beta_1 -
beta_0)*i / protocolLength);
150       double da = (a_1 - a_0) / protocolLength;
151       double dbeta = 1- 2.5545 * sin ((7*M_PI/2)*(a_0+(i+1)*da)) - (1- 2.5545 * sin
((7*M_PI/2)*(a_0+i*da))) ; // (beta_1 - beta_0)/ protocolLength;
152
153       double z_1_tmp = z_1;
154       double z_2_tmp = z_2;
155
156       double prefactor = beta*k*(a_1 - a_0) / (2 * protocolLength);
157       double dw1 = prefactor*z_1_tmp*z_1_tmp;
158
159       double dQ1 = (a*k*z_1_tmp*z_1_tmp+m*z_2_tmp*z_2_tmp)*dbeta/2; //(beta_1 -
beta_0) / (2 * protocolLength);
160       //std::cout << ((j-1)*ROI[l]+(i-1)) << "\t" << dw1 << "\t" << dQ1 <<
std::endl;
161       double tmp = dw1 + dQ1;
162
163       work += tmp;
164
165       if (l==1) {
166         energy_slow[i]+= tmp;
167       }
168       if ((l==1) && (j==NumberOfPaths) && (i==protocolLength)){
169         FILE *etslow = NULL, *etslowmgl = NULL, *etslowL = NULL;
170         etslow = fopen("energy_slow.dat", "w+"); // for saving E(t) fast
171         for (int t = 1; t <= protocolLength; t++) {
172           energy_slow[t]=energy_slow[t]/NumberOfPaths;
173           fprintf(etslow,"%3f\t%3e \n", t/(10.*1000000), energy_slow[t]);
174         }
175       }
176
177       if (l==0) {
178         energy_fast[i]+= tmp; //tmp_heat;
179       }
180       if ((l==0) && (j==NumberOfPaths) && (i==protocolLength)){
181         FILE *etfast = NULL, *etfastmgl = NULL, *etfastL = NULL;
182         etfast = fopen("energy_fast.dat", "w+"); // for saving E(t) fast
183         for (int t = 1; t <= protocolLength; t++) {
```

```
184           energy_fast[t]=energy_fast[t]/NumberOfPaths;
185           fprintf(etfast,"%3f\t%3e \n", t/(10.*1000000), energy_fast[t]);
186         }
187       }
188     }
189
190     EndDistr[j]=(z_1);
191     EndDistr_v[j]=(z_2);
192
193     if ((j == NumberOfPaths)){
194       FILE *distr = NULL, *distrmgl = NULL, *distrL = NULL;
195       std::string distring = "distribution_run"+ std::to_string(l+1) +".dat";
196       distr = fopen(distring.c_str(), "w+"); // for saving the initial distr
197       for (int i = 0; i < NumberOfPaths; i++) {
198         fprintf(distr,"%3e\t%3e\t%3e\t%3e \n", initialDistr[i], initialDistr_v[i],
EndDistr[i], EndDistr_v[i]);
199       }
200     }
201
202     meanWork += work / NumberOfPaths;
203     freeEnergy += exp(-work) / NumberOfPaths;
204     forLinearResponse += work/NumberOfPaths - work*work/(2*NumberOfPaths);
205   }
206   std::cout << "RUN " << l+1 << " ANALYZED" << std::endl;
207   freeEnergy = -log(freeEnergy);
208   double expectedFreeEnergy = log(beta_1/beta_0)+0.5*log(a_1/a_0); // free energy
using the partition function
209   th.expectedFreeEnergy = expectedFreeEnergy;
210   th.freeEnergy = freeEnergy;
211   th.inverseTau = 10*1000000 /(ROI[l]);
212   th.meanWork = meanWork;
213   th.forLinearResponse = forLinearResponse;
214   return th;
215 }
216
217 int main() {
218   int n=9;
219   vector<thermo_type> p(n);
220   vector<double> linearResponse(n);
221   std::string ofilename("Jarzynski.dat");
222   ofstream ofile(ofilename.c_str());
223   ofile << "1/tau\tFreeEnergy\tWSE\tWork\tLinearResponse" << std::endl;
224
225 #pragma omp parallel for             // for using multiple computer cores at once
226   for (int i = 0; i < n; i++) {
227     p[i] = Jarzynski(i);
228   }
229   for (int i = 0; i < n; i++){
230     linearResponse[i] = p[i].forLinearResponse + p[i].meanWork*p[i].meanWork / 2;
231   }
232   cout << "1/tau\tFreeEnergy\tWSE\tWork\tLinearResponse" << std::endl;
233   for (int i = 0; i < n; i++){
234      ofile << p[i].inverseTau << "\t" << p[i].expectedFreeEnergy << "\t" <<
p[i].freeEnergy << "\t" << p[i].meanWork << "\t" << linearResponse[i] << "\t" <<
std::endl;
235      cout << p[i].inverseTau << "\t" << p[i].expectedFreeEnergy << "\t" <<
p[i].freeEnergy << "\t" << p[i].meanWork << "\t" << linearResponse[i] << "\t" <<
std::endl;
236   }
237
238   return 0;
239 }
```

D. Deutsche Zusammenfassung

Die Anwendungen der stochastischen Thermodynamik erstrecken sich von der Biologie über die Materialwissenschaften bis hin zur Informationstheorie. Ein prominentes Beispiel ist die Jarzynski-Gleichung, die das Verhalten eines mechanischen Systems das weit vom thermodynamischen Gleichgewicht entfernt ist mit der freien Energie im Gleichgewicht in Beziehung setzt. Diese wurde zum Beispiel verwendet um die Faltungseigenschaften von Nukleinsäuren und Proteinen zu beschreiben.
Kürzlich hat die theoretische Untersuchung schneller Temperaturänderungen (thermisches Treiben) eine Verallgemeinerung der Jarzynski-Gleichung über eine rein mechanische Kontrolle hinaus ermöglicht. Trotz der Bedeutung von Prozessen, die sowohl thermische als auch mechanische Veränderungen verwenden, wie z.B. in Wärmekraftmaschinen, stehen experimentelle Untersuchungen von Fluktuationtheoremen noch aus, die auch thermisches Treiben berücksichtigen.
Das zentrale Ziel dieser Arbeit ist es diese Lücke zu füllen und die William-Searles-Evans-Gleichung und die verallgemeinerte Jarzynski-Gleichung experimentell zu untersuchen. Um dies zu erreichen, verwenden wir levitierte Nanoteilchen als einen Prüfstand, der schnelle Kontrolle sowohl über das mechanische Potential als auch ein effektives thermisches Bad ermöglicht, das über eine Rückkopplungssteuerung implementiert wird.
Ausschlaggebend für diese Studie war die Erweiterung eines bestehenden Experiments, das eine optischen Falle in einer auf einem photonischen Kristall basierenden Hohlkernfaser realisiert: Es wurde eine programmierbare, zeitabhängige Steuerung der zentralen experimentellen Parameter implementiert und ihr Einfluss auf das Detektionssystem und die Datenanalyse analysiert. Zur Auswertung der großen Datenmenge, die für diese Art von Experimenten gesammelt wird, wurde durch Parallelisierung an einem Mulit-Core-Rechner eine schnelle Datenverarbeitung ermöglicht.
In dieser Masterarbeit haben wir erfolgreich die Gültigkeit des William-Searles-Evans und des verallgemeinerten Jarzynski-Flukutationstheorems für thermisches und mechanisches Treiben weit vom thermodynamischen Gleichgewicht demonstriert für eine Rate der Zustandsänderung von bis zu zwei Größenordnungen über den Gültigkeitsbereich der quasistatischen Approximation hinaus.