Temperature measurement of ^{24}Mg atoms in a MOT using time of flight measurement

Ludwig Hruza
ETH Zürich, hruzal@student.ethz.ch

June 20, 2018

Abstract

This is a guide on how to apply the time-of-flight measurement method to determine the temperature of atoms in a MOT using my python code 'analyze_cloudpicture.py': First an expanding atom cloud is simulated at different times t. Then these images are fitted to a 2d-Gaussian fit function and the temperature of the atom cloud is obtained from the fit parameters.

Contents

1 Introduction

The aim of my project is to measure the temperature of a cloud of magnesium atoms captured in a magneto-optical trap (MOT). A MOT traps atoms by means of Doppler cooling together with a quadrupole magnetic field. Due to the Zeeman effect the degenerate energy levels of the magnesium atoms split up in such a way that the radiative force confines the atoms around $B = 0$. A description of how a MOT works can be found in [1]. In the present experiment carried out in the *Trapped Ion Quantum Information Group* at ETH Zürich the MOT is used to provide a reservoir of cold atoms, which are then used to load an optical trap. Since the trap depth is of the order of a few millikelvin, it is important to determine the temperature of the MOT.

The idea of behind the temperature measurement using a time-of-flight (Tof) measurement method is the following: Upon turning off the trap at time t=0 atoms will move into arbitrary dimensions with a mean kinetic energy that is connected to the cloud's temperature T by the equipartition theorem with three degrees of freedom

$$\langle E_{\mathrm{Kin}}\rangle = \frac{1}{2}m(\langle v_x^2\rangle + \langle v_y^2\rangle + \langle v_z^2\rangle) = \frac{3}{2}k_{\mathrm{B}}T. \quad (1)$$

Here v_i is the mean absolute velocity of all atoms in dimension i ($i = x, y, z$). Assuming isotropic velocity distribution (i.e. $(\langle v_x^2\rangle + \langle v_y^2\rangle + \langle v_z^2\rangle) = \bar{v}^2$) this is equivalent to $\bar{v}^2 = k_B T/m$. Hence the temperature T can be calculated by measuring $\bar{v}$, which is found by comparing the expansion of the cloud after different flight times t.

The theoretical derivation is presented in section 2. The main part of my work was to write a program that fits a sequence of images of an expanding cloud to the model derived in section 2 and extracts the temperature. This was done using Python and is explained in section 4. At the time when I was writing the program, no data from our MOT was available, so I implemented a simulation of an expanding falling cloud. The results are presented in section 5.

2 Theory of time-of-flight measurement

In this section I will derive how the temperature T depends on the expansion of the cloud.

If we assume that the MOT beams have a Gaussian intensity distribution we may then assume the atom cloud at $t = 0$ to have a Gaussian distribution, too with an initial standard deviation $\sigma_{i,0}$ in the corresponding dimension $i = x, y, z$. We choose the coordinate system such that the distribution is centered at position $\mathbf{r} = 0$. The probability to find an atom in a small volume dr^3 around $\mathbf{r}$ when $t = 0$ is

$$f(\mathbf{r}, t = 0) \propto \prod_{i=x,y,z} e^{-(r_i)^2/2\sigma_{i,0}^2}. \tag{2}$$

At a later time t each individual atom with initial velocity $\mathbf{v}$ will have moved w.r.t its initial position $\mathbf{r}_0$ by an amount $\mathbf{v}t$. So we are tempted to modify (2) by

$$f(\mathbf{r}, \mathbf{v}, t > 0) \propto \prod_{i=x,y,z} e^{-(r_i - v_i t)^2/2\sigma_{i,0}^2}. \tag{3}$$

However, not every velocity $\mathbf{v}$ is equally likely to occur. Hence in (3) we should have respected the normalized velocity distribution $g(v_i, T)$ (Maxwell-Boltzmann distribution), which is the probability that a non interacting atom at temperature T has a certain velocity in dimension i. Later I will use an algorithm to fit an image of the spatial distribution of atoms to the function we are about to derive. To have more degrees of freedom in this fit, I will assume that the temperature in each dimension T_i in principle could be different, i.e $g(v_i, T) \to g(v_i, T_i)$, even though in the experiment the different temperatures would thermalize. This will make the fitting algorithm more robust. When interested in the actual temperature T I will take the average of the T_i's. The spatial distribution at a later time is now obtained by integrating over all possible velocities

$$f(\mathbf{r}, t) \propto \prod_{i=x,y,z} \int_{-\infty}^{+\infty} \mathrm{d}v_i \; e^{-(r_i - v_i t)^2/2\sigma_{i,0}^2} g(v_i, T_i). \tag{4}$$

What is $g(v_i, T_i)$? Here I will give an outline of the derivation: Assume the setup to be one dimensional and divide velocity space into k intervals of lengths Δv, each interval carrying an index $j = 1, ..., k$ and yet unknown number n_j of particles with velocity v_j and hence energy $E_j = 1/2 \; m v_j^2$. It is assumed that the system of $N = \sum_{j=1}^{k} n_j$ particles takes the state $\{n_1, ..., n_k\}$ which has the biggest number of different ways one can assemble it. This state $\{n_1, ..., n_k\}$ then is obtained by maximizing the different ways one can assemble it, $P(n_j) = \frac{N!}{n_1! ... n_k!}$, with the constraints $N = \sum_{j=1}^{k} n_j$ and $E_{tot} = \sum_{j=1}^{k} n_j E_j$. Having done the maximization, the normalized velocity distribution is just $g(v_j, T) = n(v_j) := n_j$[1]. From the maximization there are two yet unknown Lagrange multipliers, which can be determined by the

[1]Note that v_j is not a velocity component in dimension j, but just one specific velocity of our one dimensional discretized velocity space.

normalization condition $\int_{-\infty}^{\infty} f(v)dv = 1$ and $\langle E\rangle := \int_{-\infty}^{\infty} \frac{1}{2}mv^2 f(v)dv = \frac{1}{2}k_BT$ (by the equipartition theorem). One finds

$$g(v,T) = \left[\frac{m}{2\pi k_B T}\right]^{1/2} \exp\left(-mv^2/2k_BT\right). \tag{5}$$

Until now the influence of gravity on the expanding atom cloud (in r_z dimension) was neglected. We fix this by substituting $r_z \to r_z + gt^2/2$ in (4). This works because gravity does not affect the evolution of the cloud's radius (which I define to be the standard deviation $\sigma_i(t)$ of the spatial distribution $f(\mathbf{r},t)$) but merely shifts the center of the cloud. Inserting (5) into (4) and performing the integration over v_x, v_y, v_z[2]

$$f(\mathbf{r},t) \propto e^{-x^2/2\sigma_x(t)^2} e^{-y^2/2\sigma_y(t)^2} e^{-(z+gt^2/2)^2/2\sigma_z(t)^2}, \tag{6}$$

where

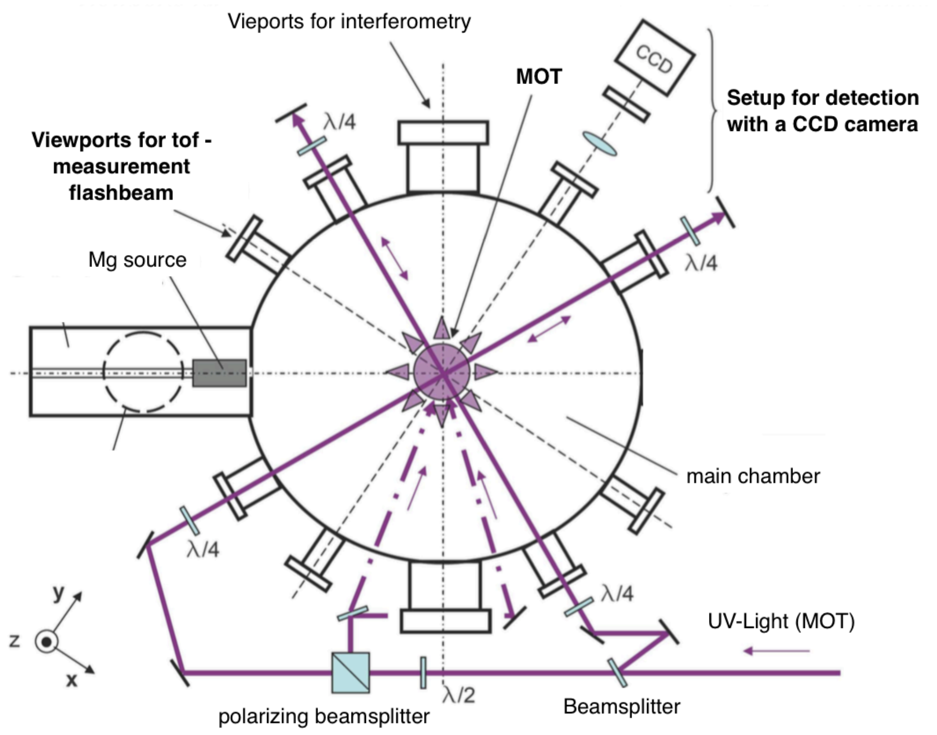
$$\sigma_i^2(t) = \sigma_{i,0}^2 + \frac{k_B T_i}{m_{Mg}} t^2. \tag{7}$$

Hence, the task is to fit the obtained images of the expanding cloud after time t to a 2d-Gaussian distribution with standard deviation $\sigma_i(t)$.

3 Experimental setup

A schematic of the experimental setup is shon in Fig 1. The cloud of ^{24}Mg atoms inside the MOT chamber is observed through a viewport of diameter 16 mm using a CCD camera[3]. The CCD camera is oriented in y direction. For some realistic initial configuration, e.g. a MOT-temperature of $T = T_i = 10$ mK and an initial radius of 0.7 mm, according to our model of an expanding cloud in (6) and (7) the cloud would take about $t_{\max} := 4.3$ ms until its radius $\sigma(t)$ has expanded beyond the field of view of the CCD camera. So we have to keep in mind, that the geometrical constraints of the viewport only allow an observation of an expanding cloud for a very short time. The resonance beam passes through another view port in x direction, i.e. orthogonal to the CCD camera. It is resonant with the $3^1S_0 \to 3^1P_1$ transition of the magnesium atoms and has the same diameter as the view port.

[2]for better readability I will from now on use the simpler notation $r_x, r_y, r_z \to x, y, z$.
[3]The model used is pco.ultraviolet.



Vieports for interferometry
MOT
CCD
Setup for detection with a CCD camera
Viewports for tof - measurement flashbeam
λ/4
Mg source
λ/4
main chamber
λ/4
λ/4
UV-Light (MOT)
y
z
x
λ/2
polarizing beamsplitter
Beamsplitter

Figure 1: Schematic of the experimental setup. Relevant parts are labeled bold. Image from [2].

4 How the code works

The code 'analyze_cloudpicture.py' consists of two parts: simulation and fitting.

Simulation is only needed to check how well the fitting algorithm works, because no data from our MOT was yet available. It works as follows: Since the CCD will capture a two dimensional image in the xz-plane and I assume that the flash beam will cause all atoms in the MOT to florescence with the same intensity, an image taken at time t should correspond to a plot of (6) integrated over all y. Since (6) is a Gaussian, the resulting function will be again a Gaussian

$$f(x,z,t) \propto e^{-x^2/2\sigma_x(t)^2} e^{-(z+gt^2/2)^2/2\sigma_z(t)^2}. \tag{8}$$

Furthermore one could imagine that a real cloud may not expand completely spherically symmetric but (due to some imbalances in the environment) a

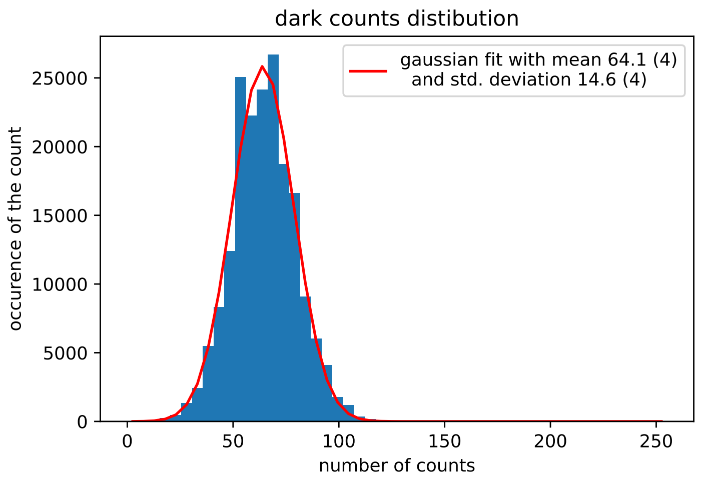


Figure 2: Histogram of a dark image take with our CCD. The CCD image can be converted to a matrix with dimensions 374×500 and the maximum pixel count is 255. The histogram is fitted with a Gaussian with mean 46.1 (4) counts and standard deviation 14.6(4). The rather great error on the mean is because the Gaussium was fitted to the bins of the histogram and not the 374×500 pixels in the first place.

bit faster in one direction than another. This is accounted for by allowing different temperatures T_x and T_z as well as a correlation $\rho := \mathrm{cor}(x, z) = \mathrm{cov}(x, z)/(\sigma_x \sigma_z)$ between x and y, such that the function modeling a cloud at time t becomes a 2d bivariate Gaussian

$$f(x, z, t) = Ae^{-\left(\frac{(x-x_0)^2}{2\sigma_x(t)^2}+\frac{(z-z_0+gt^2/2)^2}{2\sigma_z(t)^2}-\frac{2\rho(x-x_0)(z-z_0)}{\sigma_x(t)\sigma_z(t)}\right)/2(1-\rho^2)}, \tag{9}$$

where the amplitude A is chosen to equal the maximal pixel count of the CCD camera, which is 255 counts per pixel, and $(x_0,\, z_0)$ is the initial position, which until now we have set to the origin. This function is now plotted on a grid of xz-values at different times t with initial data that should correspond to our real MOT (e.g. $T_i = 10$ mK and $\sigma_{i,0} = 700\ \mu$m). Then I added noise to each of the plots according to a dark image taken with the CCD camera. A histogram of this dark image is shown in Fig. 2. Consequently the noise was modeled as a Gaussian with the corresponding mean of the dark image and a tunable standard deviation σ_{noise}. To allow the possibility of saturated images, I included an option for saturation, which increases the amplitude A above 255 counts per pixel and then clips all values above

255 back to 255. For testing the algorithm I generated a sequence of images which simulate an expanding cloud with given initial condition after different expansion times t. An example of such a sequence is shown in Fig. 3.

In the fitting part of my code a sequence of noisy and saturated cloud (such as the right column of Fig 3) is fitted back to (9), where A, x_0, z_0, $\sigma_x(t)$, $\sigma_z(t)$ and ρ are now free fitting parameters. The information about the temperature is contained in $\sigma_x(t)$ and $\sigma_z(t)$ via (7) with $T = \frac{1}{2}(T_x + T_z)$. Introducing a constant offset as a further fitting parameter to (9) (to account for the dark counts) has shown to improve the fitting a lot. Another improvement for the fitting of saturated images is achieved by excluding all pixels from the data that have a pixel count equal to the maximal pixel count of 255.

5 Results

Fig. 4 (a)-(c) show the fits to the simulated images shown in the right column of Fig. 3. They give a first evidence that the fitting part works quite well. However, when the information about $\sigma(t)$ from Fig. 4 (a)-(c) is plotted in (d) the error bars are much too small to explain the deviation from the fit line (also the case in Fig. 5). The errors calculated from my Python code stem from a in-built Python function which returns the statistical variance of the "curve_fit" algorithm, usually implemented as a least square problem. A reason for the discrepancy with the error bars could be that my fitting function has more parameters than there are physical degrees of freedom for an atom cloud with definite temperature and hence isotropic velocity distribution. E.g. the fitting function includes an offset, different standard deviations in x and z dimension and a correlation between x and z. However, these additional parameters are needed, to make sure that Python fitting algorithm "curve_fit" converges reliably.

Next I considered how well the fitting works for sequences of clouds with different observation times. In section 3 we defined the maximal observation time $t_{\max} = 4.3$ ms as the time at which the maximal field of view can capture one standard deviation of the spatial distribution of the cloud. One may wonder whether a field of view of one standard deviation is already to small to have a precise estimate for the cloud's radius. This is indeed the case as seen by comparing Fig. 5 (a) to (b).

To sum up, one should aim to record images before $t_{\max}/2$ (which is approximately when the cloud's radius $\sigma(t)$ has expanded to half the field of view) and rethink how to define sensible error bars.

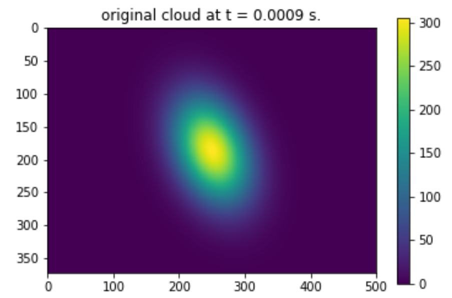
6 Acknowledges

I would like to thank my supervisor, Christoph Fischer, for his support and his patient explanations on all kinds of MOT related questions and especially for fixing the mirrors if I had messed up with it (I had started to experimentally realize the flash beam, but due to problems with our acusto-optical modulator this did't lead to results in the time available).

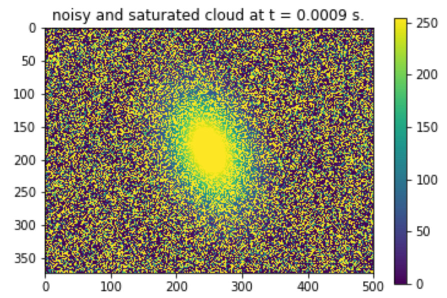
References

[1] J. C. Foot. *Atomic Physics.* Oxford University Press, 2005.

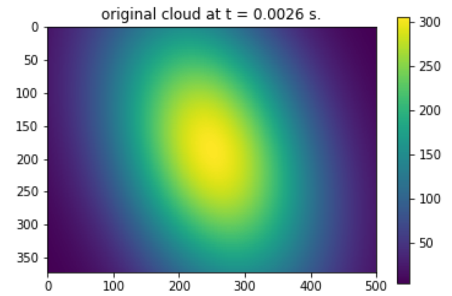
[2] Nils. Rehbein. Realisierung neuer laserkühlverfahren und spektroskopielaser für einen optischen magnesium-frequenzstandard, 2006.



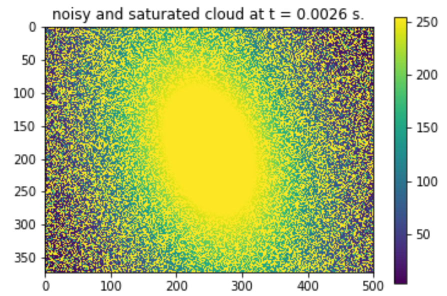
(a)



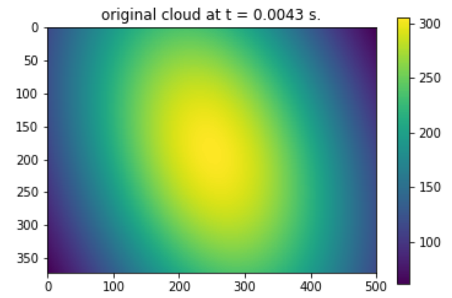
(b)



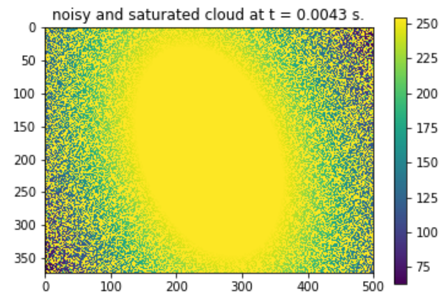
(c)



(d)

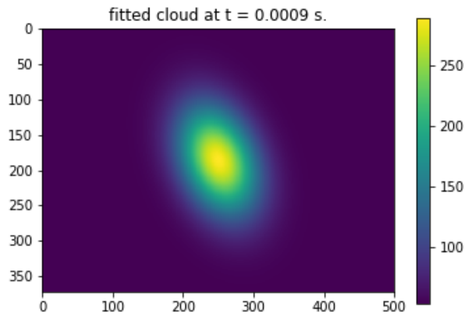


(e)

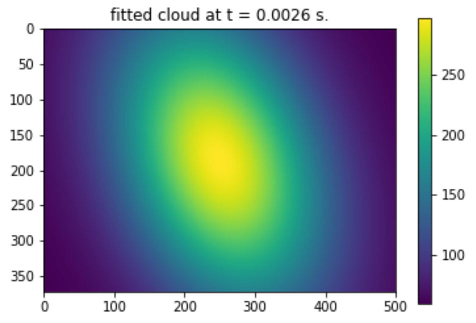


(f)

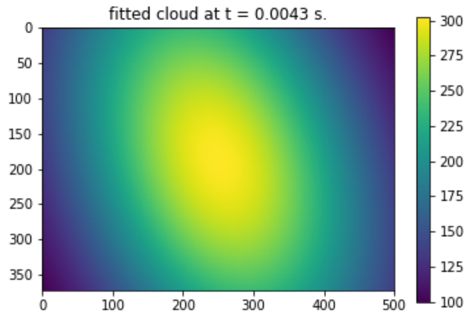
Figure 3: Left column: Secuence of clouds with initial values $T_x = 10$ mK, $T_z = 20$ mK, $\rho = 0.3$ and $\sigma_{i,0} = 700$ μm. The saturation level is $A/(\text{max. pixel count}) = 1.2$ and no noise is added yet. Right column: noise is added to the left column with a signal to noise ratio $(\text{maximum signal})/(2\sigma_{\text{noise}}) = 0.5$.



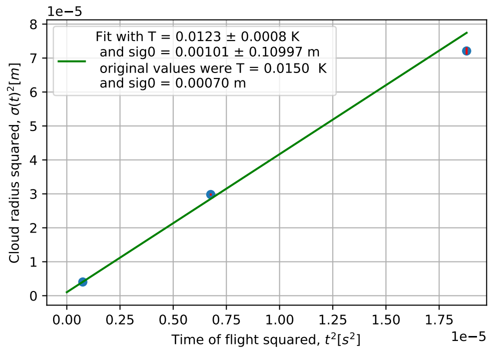
(a)



(b)

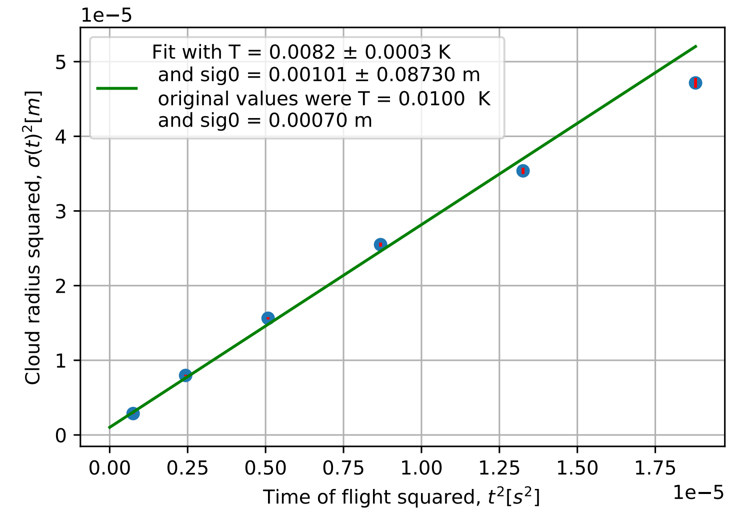


(c)

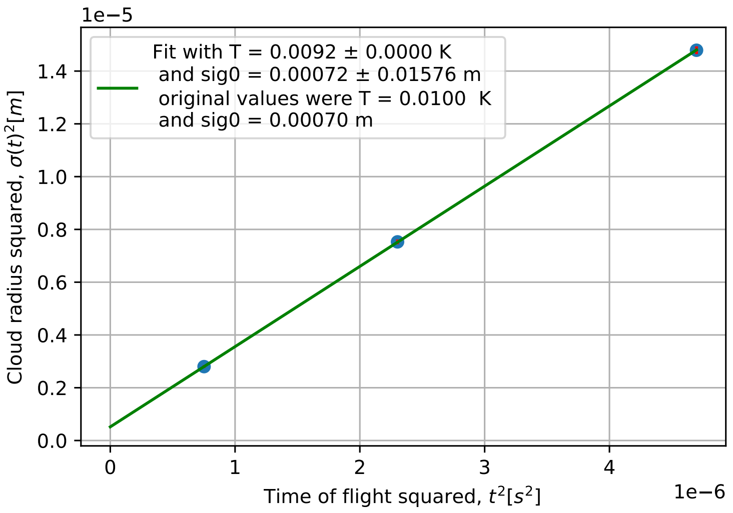


(d)

Figure 4: (a)-(c): Plot of a sequence of clouds with parameters obtained from a fit of Fig. 3 (b),(d),(f). (d): $\sigma(t) = (\sigma_x^2(t) + \sigma_y^2(t))/2$ as obtained from (a)-(c). It should have a linear dependency on t according to (7). The fit (green line) gives a temperatur of 12.3 (0.3) mK, while the original temperature was $T = (10 + 20)/2$ mK = 15 mK. (See Fig. 3). The error bars calculated by my algorithm still seem to be far to small.



(a)



(b)

Figure 5: Result of fitting for initial values $T_x = T_z = 10$ mK, $\rho = 0$ and $\sigma_{i,0} = 700$ μm. (a) represents 6 data points, corresponding to 6 images taken evenly spaced between $t_{\max}/5$ and $t_{\max}$. For (b) only three images were taken evenly spaced between $t_{\max}/5$ and $t_{\max}/2$.