Diplomarbeit

Numerical simulation of radio-frequency dressed potentials for Caesium atoms

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

eingereicht an der Fakultät für Physik der Technischen Universität Wien

von

Christian Altvater

Matrikelnummer 01607332

unter der Anleitung von

Univ.Prof. Dipl.-Phys. Dr.rer.nat. **Thorsten Schumm**
Univ.Ass. Dipl.-Phys. Dr.rer.nat. **Stephanie Manz**
Univ.Ass. Dipl.-Ing. **Benedikt Gerstenecker**, BSc

Atominstitut
Quantenmetrologie
Technische Universität Wien
Stadionallee 2, 1020 Wien, Österreich

Wien, Februar 2022

Abstract

This thesis provides a numerical simulation of radio-frequency (RF) dressed potentials of Caesium atoms and the control software of a Tabor WW5062 dual-channel waveform generator.

For the first part, an effective magnetic field is calculated from a combined static and oscillating field. The Biot-Savart law yields the static field and the RF field amplitudes. By applying unitary transformations in form of rotation matrices and the rotating wave approximation (RWA), the Hamiltonian can be simplified and solved for the potential energy states. The validity conditions of the RWA are checked and visually illustrated. The results of the potential calculation can be displayed in 3D, 2D and 1D (along axes through the minimum).

The control software of the waveform generator functions as the communication between the user input to the *ADwin* experiment control interface and the device itself. The *ADwin* sends commands for the parameters, which are then set by the control software.

All of the code has been written in MATLAB R2020a.

Zusammenfassung

Diese Arbeit umfasst eine numerische Simulation von “dressed states” von Cäsium-Atomen, also von Potentialzuständen in einer Kombination aus statischem und mit Radiofrequenz (RF) oszillierendem Magnetfeld, sowie eine Steuerungssoftware für einen Tabor WW5062 Funktionsgenerator.

Das Biot-Savart Gesetz, angewandt auf die vorliegende Geometrie, erlaubt die Berechnung des statischen Feldes sowie der Amplitude des oszillierenden Feldes, aus welchen dann ein effektives Magnetfeld berechnet werden kann. Der Hamilton-Operator kann mithilfe von unitären Transformationen, umgesetzt als Rotationsmatrizen, sowie der Rotating Wave Approximation (RWA) vereinfacht und schlussendlich gelöst werden. Das Ergebnis davon sind die gesuchten Potentialzustände. Die RWA erfordert dazu die Erfüllung bestimmter Kriterien, die im Laufe der Simulation überprüft werden. Die Ergebnisse werden graphisch dargestellt. Das Ergebnis der Simulation wird in 3D für eine Äquipotentialfläche sowie in 2D und 1D jeweils entlang der Achsen durch das Potentialminimum abgebildet.

Die Steuerungssoftware für den Funktionsgenerator kann als Kommunikation zwischen den Nutzereingaben in die *ADwin* Experimentsteuerung und dem Gerät selbst gesehen werden. Die Experimentsteuerung sendet die gewünschten Parameter an die Software, die diese dann umsetzt.

Der Code für diese Arbeit wurde vollumfänglich in MATLAB R2020a geschrieben.

Contents

1 Introduction

The concept of Bose-Einstein condensates (BECs) has existed for almost 100 years now, since 1924 [1, 2]. A BEC is reached by cooling the Bose gas into the ground state of a confining potential. In 1995, 70 years after its theoretical description, the first-ever BEC produced in an experiment was reached using ^{87}Rb atoms [3]. This breakthrough founded a completely new field of research: ultracold atoms. Achieving condensation for ^{133}Cs is more complex than for other alkali atoms due to the necessary control of its scattering properties. About eight years later, in 2003, the first BEC made of ^{133}Cs was accomplished in Innsbruck. [4]

Trapping atoms is of fundamental importance for Bose-Einstein condensation. Static magnetic traps are a natural choice for that purpose, but Maxwell's equations limit them to single-well field minima only. In 2001, a new method for magnetic trapping of atoms was proposed by Zobay and Garraway: When coupling different spin states by a resonant driving field, Maxwell's equations don't apply anymore and more complex potential shapes like double-wells are possible. The resulting potential states are called *dressed potentials*. [5]

To get a better understanding of ultracold Caesium atoms, learning about their behavior in various trap configurations is important. This thesis provides a numeric simulation of radio-frequency (RF) dressed potentials of Caesium atoms for a specific atom chip geometry and the control software of a dual-channel waveform generator to review the results in practice.

2
Theory

2.1 The Caesium atom

The experiment on which this thesis is based uses^{133}Cs, the only stable Caesium isotope. The following chapter refers to ^{133}Cs only. Since the creation of a BEC is out of the scope of this thesis, it will only give general information about the Caesium atom to put the motivation of the whole experiment into perspective.

2.1.1 Physical and optical properties

Caesium is an alkali atom, meaning it has a single valence electron. Its nuclear spin is $7/2$ and its electron spin, resulting from the only valence electron, is $1/2$. Due to its integer overall spin, ^{133}Cs is considered a boson. The recoil temperature, or in other terms the energy given to the atom by scattering a single photon, indicates the minimum temperature achievable with laser cooling. It is only 200 nK for ^{133}Cs due to its large mass of 133 amu.

The ground state of ^{133}Cs can be written as $6\,^2S_{1/2}$. The D_1 line ($6\,^2S_{1/2} \rightarrow 6\,^2P_{1/2}$) has a frequency of $2\pi \cdot 335$ THz with a natural line width of $\Gamma_1 = 2\pi \cdot 4.56$ MHz. The frequency of the D_2 line ($6\,^2S_{1/2} \rightarrow 6\,^2P_{1/2}$) is $2\pi \cdot 352$ THz, its natural line width is $\Gamma_2 = 2\pi \cdot 5.22$ MHz. The D_2 line is often used for cooling and imaging Caesium atoms. The hyperfine splitting of the ground state between the states F = 3 and F = 4 stands at 9.2 GHz, which is used for the SI definition of the second. The hyperfine splitting of the $6\,^2P_{1/2}$ excited state lies between 150 MHz and 250 MHz. The hyperfine structure of both Caesium D lines is shown in figure 2.1. [6]

2.1.2 Scattering properties

The path to Bose-Einstein condensation involves atom trapping and evaporative cooling. Scattering properties of the used atoms determine the effectiveness of the latter.

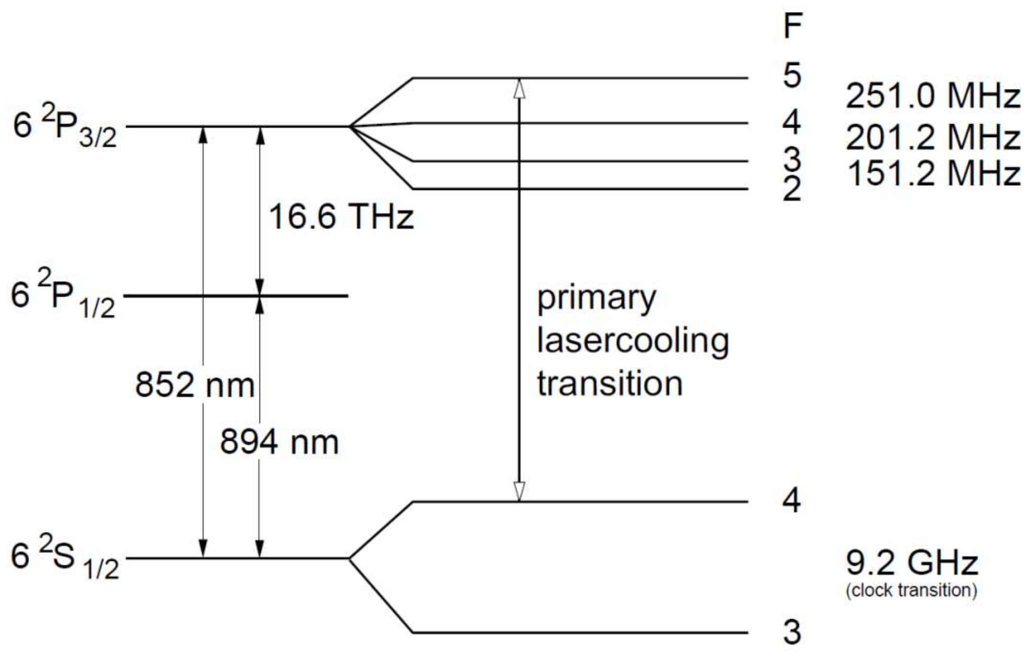


Figure 2.1: ^{133}Cs D lines [4]

Inelastic scattering
Inelastic collisions can convert potential energy into kinetic energy (exothermic) or vice versa (endothermic). Endothermic collisions can usually be excluded for ultracold atoms at temperatures of a few µK. Exothermic collisions, on the other hand, raise the kinetic energy of the atoms, which results in heating or losing atoms from the trap. One of the main contributions to the inelastic scattering of Caesium atoms is the dipolar interaction, which can be strong for heavy atoms [4].

Elastic scattering
Elastic collisions transfer kinetic energy between the involved particles. In the present context of condensation, they are fundamentally important for evaporative cooling, since they enable thermalization of the atom gas.

Scattering length
The scattering length is an important parameter for making Caesium BECs. In particular, the elastic s-wave scattering length is inherently negative, indicating an attraction between atoms, which impedes thermalization. However, by adding an external magnetic field it can be tuned to a reasonable value. This relation is shown in figure 2.2 for Caesium in the ground state $F = 3, m_F = 3$. Its zero-crossing is at 17 G, a usual value for trapping and evaporative cooling is 300 a_0 at 23 G, where a_0 denotes Bohr's radius. Furthermore, a Feshbach resonance can be found at 48 G.

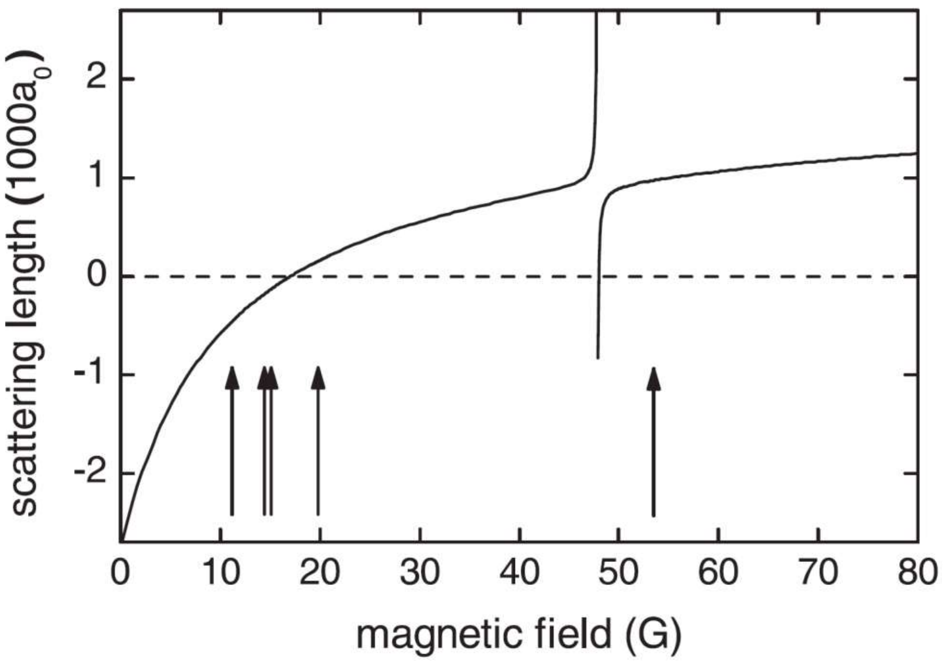


Figure 2.2: Scattering length as a function of the external magnetic field for Caesium in the ground state $F = 3, m_F = 3$. Arrows indicate narrow Feshbach resonances [7].

2.2 Atoms in static magnetic fields

The content of the following chapter 2.2 is summarized mainly from [8].

The energy state of an atom is determined by the quantum numbers n, l and F. In the absence of external magnetic fields, it is degenerate in the magnetic quantum number m_F. By applying an external field $\mathbf{B}(\mathbf{r})$, this degeneracy is removed due to an interaction of the field with the magnetic moments of the atom's electrons and nucleus (for alkali atoms like Caesium, only the valence electron contributes). The magnetic moments arise from the total electron angular momentum $\mathbf{J}$ and the nuclear angular momentum $\mathbf{I}$.

2.2.1 Linear Zeeman effect

The splitting of energy levels in the presence of a magnetic field is called Zeeman effect. Since only small fields on the order of 10 G are needed for trapping atoms, the energy splitting is linear, hence the name linear Zeeman effect (for strong field strengths, it is not linear anymore and called Paschen-Back effect). The Zeeman energy is given by

$$H_{Zeeman} = -\boldsymbol{\mu}_F \cdot \mathbf{B} = g_F \mu_B \mathbf{F} \cdot \mathbf{B}. \tag{2.1}$$

μ_F is the total magnetic moment of the current state, g_F the corresponding Landé factor and $\mu_B = 9.274 \cdot 10^{-24} J/T$ Bohr's magneton. $\mathbf{F} = \mathbf{I} + \mathbf{J}$ is the total atomic angular momentum. This linear approximation is only valid if the energy shift is small compared to the hyperfine splitting.

2.2.2 Adiabatic approximation

The motion of a neutral atom in a static magnetic field can be described by the Hamiltonian

$$H = E_{kin} + H_{Zeeman} = \frac{\mathbf{p}^2}{2m} + g_F \mu_B \mathbf{F} \cdot \mathbf{B}(\mathbf{r}). \tag{2.2}$$

In order to solve equation 2.2, it is necessary to diagonalize the interaction of the magnetic field $\mathbf{B}(\mathbf{r})$ with the total angular momentum $\mathbf{F}$. This transforms equation 2.2 into an eigenvalue equation, where the eigenvalues represent the atomic energy levels. The Hamiltonian H can be diagonalized by applying the unitary transformation U_S, which aligns the magnetic field with the quantization axis of $\mathbf{F}$. This usually is the z-axis in the lab frame. Aligning a vector with the z-axis consists of two steps and is shown here by rotating the vector $\mathbf{B}(\mathbf{r}) = (B_X(\mathbf{r}), B_Y(\mathbf{r}), B_Z(\mathbf{r}))$:

1. Rotation in the x-y-plane: Rotate the vector around the z-axis to align it with the x-axis. The new x-coordinate is $\sqrt{B_X(\mathbf{r})^2 + B_Y(\mathbf{r})^2}$. The rotation angle is given by

$$\alpha(\mathbf{r}) = tan^{-1}\left(-\frac{B_Y(\mathbf{r})}{B_X(\mathbf{r})}\right) - \begin{cases} 0, & \text{if } B_X(\mathbf{r}) \geq 0 \\ \pi, & \text{if } B_X(\mathbf{r}) < 0 \end{cases}. \tag{2.3}$$

   The two dimensional rotation is shown in figure 2.3.

2. Rotation in the x-z-plane: Rotate the vector around the y-axis to align it with the z-axis. The new z-coordinate is $|\mathbf{B}(\mathbf{r})| = \sqrt{B_X(\mathbf{r})^2 + B_Y(\mathbf{r})^2 + B_Z(\mathbf{r})^2}$. The rotation angle is given by

$$\beta(\mathbf{r}) = tan^{-1}\left(-\frac{\sqrt{B_X(\mathbf{r})^2 + B_Y(\mathbf{r})^2}}{B_Z(\mathbf{r})}\right) - \begin{cases} 0, & \text{if } B_Z(\mathbf{r}) \geq 0 \\ \pi, & \text{if } B_Z(\mathbf{r}) < 0 \end{cases}. \tag{2.4}$$

   The final step is shown in figure 2.4.

The additional subtraction of π is necessary to prevent an unwanted phase shift. Using these angles, the unitary transformation can be written as

$$U_S(\mathbf{r}) = exp[-iF_z\alpha(\mathbf{r})]exp[-iF_y\beta(\mathbf{r})]. \tag{2.5}$$

Since it involves two rotations, its effect on the product $\mathbf{F} \cdot \mathbf{B}(\mathbf{r})$ can be expressed as two rotation matrices $R_i(\phi)$, where the index i determines the rotation axis and ϕ specifies the angle:

$$\begin{aligned} U_S^\dagger \mathbf{F} \cdot \mathbf{B}(\mathbf{r}) &= [R_y(-\beta(\mathbf{r}))R_z(-\alpha(\mathbf{r}))\mathbf{F}] \cdot \mathbf{B}(\mathbf{r}) \\ &= \mathbf{F} \cdot [R_y(\beta(\mathbf{r}))R_z(\alpha(\mathbf{r}))\mathbf{B}(\mathbf{r})] \\ &= F_z|\mathbf{B}(\mathbf{r})|. \end{aligned} \tag{2.6}$$

The result of applying U_S to the whole Hamiltonian of equation 2.2 is

$$U_S^\dagger H U_S = U_S^\dagger \frac{\mathbf{p}^2}{2m} U_S + m_F g_F \mu_B |\mathbf{B}(\mathbf{r})| = \frac{1}{2m}[\mathbf{p} + \mathbf{A}(\mathbf{r})]^2 + m_F g_F \mu_B |\mathbf{B}(\mathbf{r})|. \tag{2.7}$$

$\mathbf{A}(\mathbf{r}) = U_S^\dagger \mathbf{p} U_S$ is the "gauge" field. Please note that usually only one m_F state is used, which replaces F_z in equation 2.7.

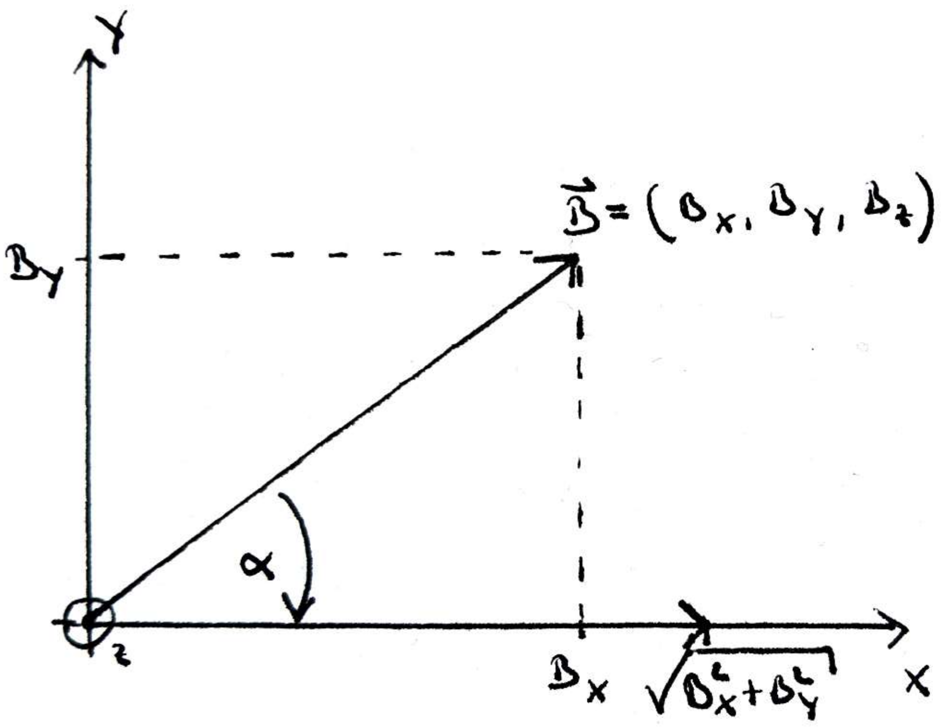


Figure 2.3: First step of aligning an arbitrary vector with the z-axis: Rotation in the x-y-plane.

In order to simplify the Hamiltonian in equation 2.7, the gauge field is neglected by applying the adiabatic approximation. Its validity is constrained by the rate of change of the magnetic field in comparison to the larmor frequency of the atomic spin:

$$\frac{\frac{d}{dt}\mathbf{B}}{|\mathbf{B}|} < \frac{g_F \mu_B}{\hbar}|\mathbf{B}(\mathbf{r})| = \omega_{Larmor}. \tag{2.8}$$

The Hamiltonian of equation 2.2 then simplifies to

$$H = \frac{\mathbf{p}^2}{2m} + V_{mag}, \tag{2.9}$$

with the potential energy

$$V_{mag} = m_F g_F \mu_B |\mathbf{B}(\mathbf{r})|. \tag{2.10}$$

2.2.3 Magnetic traps for neutral atoms

In order to create a magnetic trap, the potential energy of the atoms has to exhibit a minimum at some point in space. One can see from equation 2.10 that V_{mag} depends not only on the absolute value of the magnetic field $|\mathbf{B}(\mathbf{r})|$, but also on the product

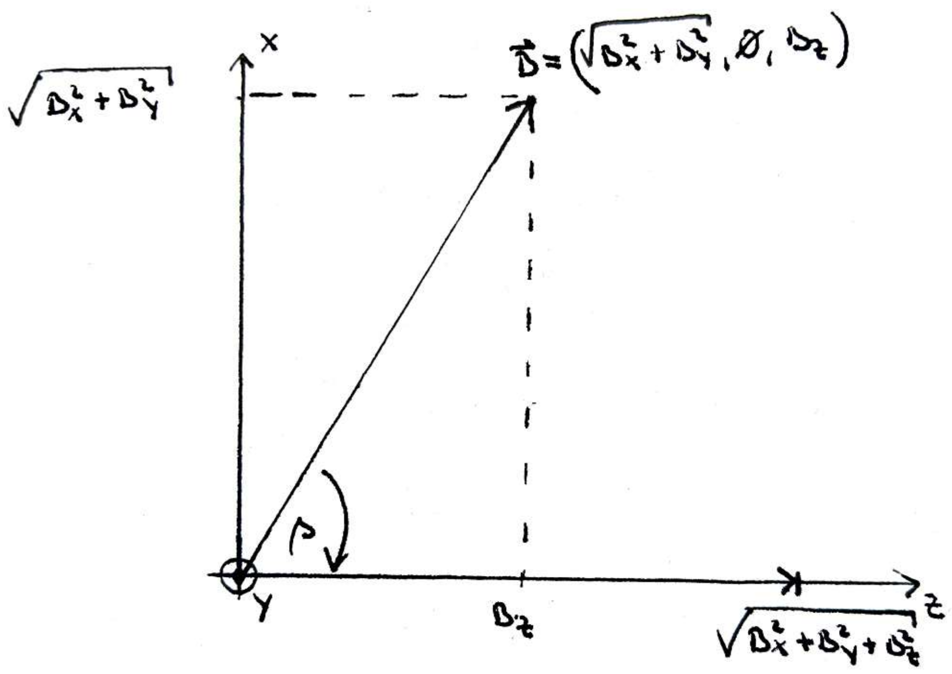

x
$\vec{B} = (\sqrt{B_x^2 + B_y^2}, \emptyset, B_z)$
$\sqrt{B_x^2 + B_y^2}$
β
y
B_z
z
$\sqrt{B_x^2 + B_y^2 + B_z^2}$

Figure 2.4: Final step of aligning an arbitrary vector with the z-axis: Rotation in the x-z-plane.

$m_F g_F$, which specifies the relative orientation of the atomic spin to the magnetic field. Consequently, the magnetic states can be split into three groups. In the case of $m_F g_F < 0$, the atoms are drawn towards regions with higher magnetic field strength and are thus called *high-field seekers*. On the other hand, *low-field seekers* are defined by $m_F g_F > 0$; these atoms are drawn towards regions of minimum magnetic field strength. If $m_F g_F = 0$, atoms are not affected by the field at all and cannot be caught in a magnetic trap.

Following Maxwell's equations, the divergence of the magnetic field in source-free space has to be zero. Hence no maxima of field strength are allowed for static fields. Local minima, on the contrary, can certainly be achieved due to the vectorial nature of the field. Therefore, it is only possible to magnetically trap low-field seekers.

2.2.4 Common trapping configurations

A magnetic trap for neutral atoms requires special field configurations yielding a local minimum. Depending on the field strength at their minimum, the traps can be separated into two types: ones with zero field at the minimum and others with non-zero field everywhere.

Quadrupole trap

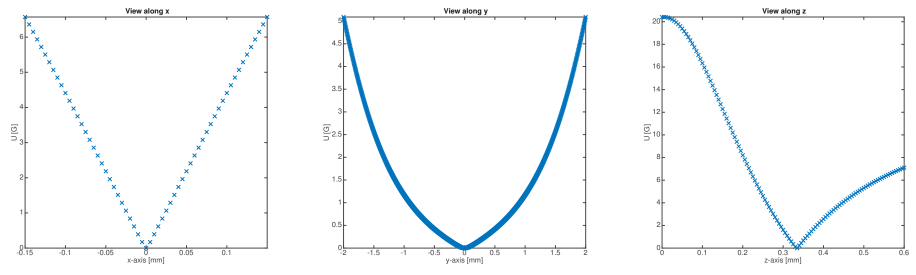
The most common trap configuration with vanishing field strength at the minimum is the quadrupole trap. Its magnetic field has the form

$$\mathbf{B} = (B'_x x, B'_y y, B'_z z), \tag{2.11}$$

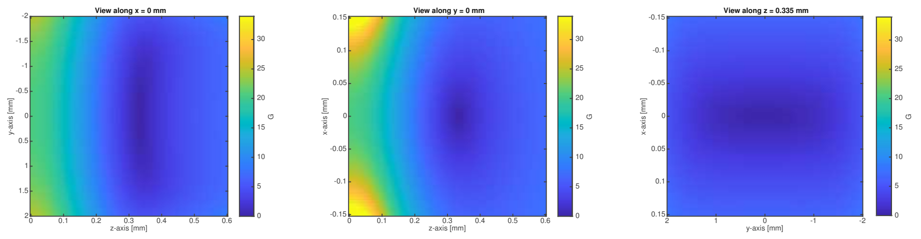
which satisfies Maxwell's equation $\nabla \cdot \mathbf{B} = 0$ if $B'_x + B'_y + B'_z = 0$.

This can be achieved with a pair of coils in "anti-Helmholtz" configuration.

In comparison to other trap geometries, the quadrupole field has the largest field gradient, which results in the strongest achievable confinement. A common problem of traps of that type is the vanishing field at the center: When there is no field at one point of the trap, the magnetic states stay degenerate and transitions between different states are possible. More precisely, low-field seeking states may be converted to high-field seekers, which are driven out of the trap. This is also called *Majorana spin-flips* [9]. The colder the confined atoms are, the bigger the problem gets, since cold atoms are located closer to the center. The shape of the magnetic field of a quadrupole trap is shown in figure 2.5.



(a)



(b)

Figure 2.5: Quadrupole trap: Magnetic field in (a) 1D and (b) 2D along axes through minimum.

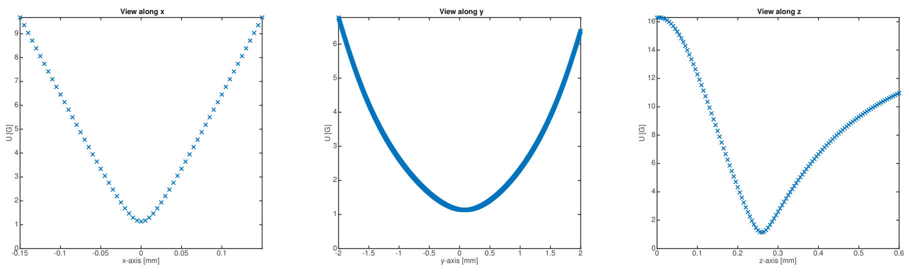
Ioffe-Pritchard traps

The vanishing magnetic field in the center of quadrupole traps poses a problem for trapping atoms. Lifting the potential is not possible without also changing its shape. This leads to a magnetic potential that can be approximated to be harmonic in the center (instead of linear for quadrupole traps), and therefore to reduced confinement of the atoms in the trap. Such a trap configuration with a non-vanishing field at the center has been proposed by Pritchard, based on the work of Ioffe for plasma confinement, and is thus called Ioffe-Pritchard-trap [10].

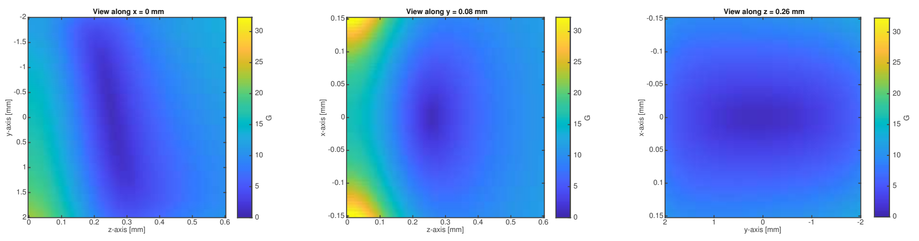
Harmonic potentials are completely determined by their oscillation frequencies, which depend on the curvature of the field along the main axes of the trap:

$$\omega_i = \sqrt{\frac{1}{m}\frac{d^2V}{dx_i^2}} = \sqrt{\frac{\mu_B m_F g_F}{m}\frac{d^2B}{dx_i^2}}. \tag{2.12}$$

The frequencies ω_i are a measure for the steepness of the potential around the center and for the extent of the harmonic area. Due to Maxwell's equations, again, no isotropic harmonic trap can be realized. The most common field configurations involve two frequencies, that are equal to each other and larger than the third one, which results in a "cigar" shaped trap (cf. figure 2.6).



(a)



(b)

Figure 2.6: Ioffe-Pritchard trap: Magnetic field in (a) 1D and (b) 2D along axes through minimum.

2.3 Radio-frequency adiabatic potentials

The content of the following chapter 2.3 is summarized mainly from [8].

As seen in section 2.2, maxima of the absolute static magnetic field $|\mathbf{B}(\mathbf{r})|$ are forbidden by Maxwell's law $\nabla \cdot \mathbf{B} = 0$. By applying an additional resonant oscillating magnetic field, separated spin states are coupled, yielding a new, combined potential shape. Since the total magnetic field is not static anymore, Maxwell's equation does not apply and more complex potential shapes like double-wells or rings are possible. The resulting states are known as *dressed potentials*.

2.3.1 Dressed state Hamiltonian of an atom in a radio-frequency field

Assume an atom in a static field $\mathbf{B}_s(\mathbf{r})$, dressed by the radio-frequency (RF) field $\mathbf{B}_{rf}(\mathbf{r})e^{i\omega_{rf}t}$ with the single frequency $\nu_{rf} = \frac{\omega_{rf}}{2\pi}$. The eigenstates of the system are now determined by the magnetic quantum number $\tilde{m}_F$ of the atom in the total magnetic field and the RF photon number N. The Hamiltonian is given by

$$H = \frac{\mathbf{p}^2}{2m} + g_F\mu_B\mathbf{F}\cdot\mathbf{B}_s(\mathbf{r}) + \hbar\omega_{rf}a^\dagger a + \frac{g_F\mu_B}{2\sqrt{\langle N\rangle}}\left[\mathbf{B}_{rf}(\mathbf{r})a^\dagger + h.c.\right]\cdot\mathbf{F}. \tag{2.13}$$

$a^\dagger$ and a are the creation and annihilation operators for quanta of the RF field:

$$a^\dagger |m_F, N\rangle = \sqrt{N+1}\, |m_F, N+1\rangle \tag{2.14}$$

$$a\, |m_F, N\rangle = \sqrt{N}\, |m_F, N-1\rangle \tag{2.15}$$

$\langle N \rangle$ is the average number of photons in the RF field.

The RF field is defined as a superposition of different phase-shifted parts

$$\mathbf{B}_{rf}(\mathbf{r}) e^{i\omega_{rf}t} = \left[\sum_k \mathbf{B}_k(\mathbf{r}) e^{i\phi_k} \right] e^{i\omega_{rf}t}. \tag{2.16}$$

The amplitude vector $\mathbf{B}_{rf}(\mathbf{r})$ is complex and has to be included in the hermitian conjugation in equation 2.13.

The first two terms of the Hamiltonian in equation 2.13 resemble the static case of equation 2.2. The third term describes the energy of the RF field, the last term is the coupling of the dressing field to the atom.

The first step for solving the Hamiltonian is applying the same unitary transformation U_S (equation 2.5). This effectively aligns the static magnetic field in each spatial point $\mathbf{r}$ with the z-axis. The RF field has been rotated as well and now contains parts that are perpendicular and parallel to the static field, respectively:

$$\tilde{\mathbf{B}}_{rf}(\mathbf{r}) = U_S^\dagger \mathbf{B}_{rf}(\mathbf{r}) U_S = \begin{pmatrix} B_{rf\perp,x} \\ B_{rf\perp,y} \\ B_{rf\parallel} \end{pmatrix}. \tag{2.17}$$

By applying a rotation U_{rf} around the z-axis, which doesn't change the static field vectors, but the perpendicular RF vectors, the field $\tilde{\mathbf{B}}_{rf}(\mathbf{r})$ can be simplified to

$$U_{rf}^\dagger U_S^\dagger \mathbf{B}_{rf}(\mathbf{r}) U_S U_{rf} = \begin{pmatrix} B_{rf\perp} \\ 0 \\ B_{rf\parallel} \end{pmatrix}, \tag{2.18}$$

which is shown in figure 2.7.

The total Hamiltonian now looks like

$$\begin{aligned} H &= E_{kin} + H_{rf} \\ &= \frac{1}{2m}\left[\mathbf{p} + \tilde{\mathbf{A}}(\mathbf{r})\right]^2 + m_F g_F \mu_B |\mathbf{B}(\mathbf{r})| + \hbar\omega_{rf} a^\dagger a \\ &\quad + \frac{g_F \mu_B}{2\sqrt{\langle N \rangle}} \left[\left(B_{rf\perp}(\mathbf{r}) a^\dagger + h.c. \right) F_x + B_{rf\parallel}\left(\mathbf{r}\right) a^\dagger + h.c.\right) F_z \right], \end{aligned} \tag{2.19}$$

with $\tilde{\mathbf{A}} = U_{rf}^\dagger U_S^\dagger \mathbf{p} U_S U_{rf}$.

In analogy to the static field case, there is a potential energy term H_{rf} that has to be solved to obtain the energy states of the atom.

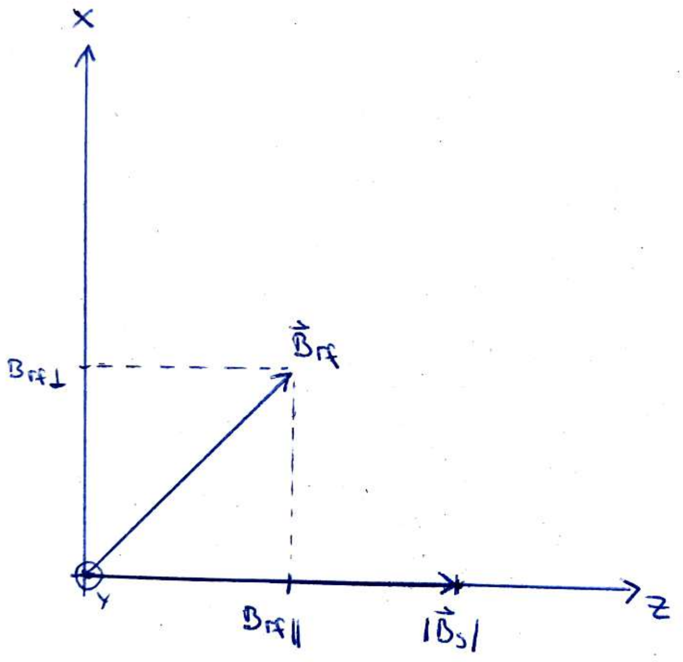


Figure 2.7: Rotated static and RF field vectors

2.3.2 Adiabatic RF potentials in the rotating wave approximation

The term in the Hamiltonian in equation 2.13 responsible for coupling the field and the atom includes resonant and off-resonant parts. The former combine the creation/annihilation of an RF photon with the simultaneous lowering/raising of the spin quantum number, while the latter describe virtual processes of lowering/raising the spin quantum number with annihilation/creation of an RF photon. In the rotating wave approximation (RWA), it is possible to simplify the Hamiltonian by neglecting the off-resonant terms. Another explanation of the RWA can be given by the polarization of the RF field. An oscillating field consists of one linearly and two circularly polarized parts. The atomic spin can be interpreted as a precession. If the system is now transferred into a frame rotating with the atomic spin, the circularly polarized component with the same rotational direction gets nearly time-independent, while the other one rotates with the sum of both frequencies. The latter one can usually be averaged to zero and neglected. The linear component is aligned with the axis of the spin precession and can be neglected if its Larmor frequency is small compared to the oscillation frequency of the field. The conditions for the RF field that have to be fulfilled to apply the RWA are thus given by:

- The detuning Δ, which is the difference of the oscillation frequency of the driving field ω_{rf} and the atomic transition frequency ω_0, has to be small compared to ω_0: $\Delta = \omega_{rf} - \omega_0 \ll \omega_0$. The off-resonant contributions to the term proportional to F_x in equation 2.19 can be neglected in that case.

- The Rabi frequency Ω, which is the transition frequency of the population of a two-state-system in a (near-)resonant driving field, has to be small compared to the Larmor frequency ω_0: $\Omega \ll \omega_0$. Otherwise, the RWA is not applicable anymore. More details on this condition can be found in [8].

- The Larmor frequency associated with the RF field component parallel to the static field has to be small compared to the RF frequency: $\left|\mu \mathbf{B}_{rf\parallel}(\mathbf{r})\right| \ll h\nu_{rf}$. This allows neglecting the term in equation 2.19 proportional to F_z.

Applying the RWA to the Hamiltonian in equation 2.19 and dropping the photon number, which can be done because the specific quantum state of the RF field is not important to this calculation, yields the final Hamiltonian

$$H_{final,RWA} = \frac{\mathbf{p}^2}{2m} + g_F\mu_B \mathbf{B}_{eff}(\mathbf{r}) \cdot \mathbf{F} = \frac{(\mathbf{p}+\mathbf{A}(\mathbf{r}))^2}{2m} + g_F\mu_B|\mathbf{B}_{eff}(\mathbf{r})|F_z, \tag{2.20}$$

with the effective magnetic field

$$\mathbf{B}_{eff}(\mathbf{r}) = \begin{pmatrix} \frac{\left|B_{rf\perp}(\mathbf{r})\right|}{2} \\ 0 \\ |\mathbf{B}_s(\mathbf{r})| - \frac{\hbar\omega_{rf}}{\mu_B|g_F|} \end{pmatrix}. \tag{2.21}$$

The eigenvalues of the Hamiltonian in equation 2.20 are, in analogy to the static field case in equation 2.10, $\tilde{m}_F g_F \mu_B |\mathbf{B}_{eff}(\mathbf{r})|$. $\tilde{m}_F$ can be interpreted as new magnetic quantum number with respect to the quantization axis defined by $\mathbf{B}_{eff}$. The energy separation of the resulting dressed states is now given by $\hbar\sqrt{\Delta^2+\Omega^2}$.

Orientation of the effective field

The vectors $\mathbf{B}_s(\mathbf{r})$ and $\mathbf{B}_{rf}(\mathbf{r})$ define a plane containing the effective field $\mathbf{B}_{eff}$. Its relative orientation to the static field is given by the angle θ:

$$tan\theta = -\frac{\Omega}{\Delta} \qquad 0 \le \theta \le \pi, \tag{2.22}$$

with the detuning $\Delta(\mathbf{r}) = |\mu_B g_F||\mathbf{B}_s(\mathbf{r})| - \hbar\omega_{rf}$ and the Rabi frequency $\Omega(\mathbf{r}) = \frac{\mu_B g_F}{2}|B_{rf\perp}(\mathbf{r})|$. For large negative detuning, θ goes towards zero and the effective field reduces to the static field. Large positive detuning, on the other hand, results in θ tending towards π, implying that the effective field becomes anti-parallel to the static field. On resonance, $\Delta = 0$, the effective field is perpendicular to the static field. Similar to the fields, the dressed states can be interpreted as rotated spin states of the static field:

$$|\tilde{m}_F\rangle = R_y(\theta)\,|m_F\rangle\,. \tag{2.23}$$

Adiabacity of the dressed states

The validity of the adiabatic approximation depends on whether the coupling of dressed states induced by the gauge term $\mathbf{A}(\mathbf{r})$ is negligible. The adiabacity condition for dressed states can be written with the angle θ and the corresponding energy separation:

$$\left|\dot{\theta}\right| \ll \sqrt{\Delta^2+\Omega^2}. \tag{2.24}$$

The states $\tilde{m}_F$ are constants of motion and can follow the effective magnetic field adiabatically if its rate of change is slow compared to the associated Larmor frequency $\sqrt{\Delta^2+\Omega^2}$.

Effective adiabatic RF potential
After neglecting the gauge field $\mathbf{A}(\mathbf{r})$ in the adiabatic approximation, the Hamiltonian of equation 2.20 yields the effective adiabatic potential

$$\begin{aligned} V_{ad,RWA}(\mathbf{r}) &= \mu_B \tilde{m}_F g_F \sqrt{\Delta^2(\mathbf{r}) + \Omega^2(\mathbf{r})} \\ &= \mu_B \tilde{m}_F g_F \left[\left(|\mathbf{B}_s(\mathbf{r})| - \frac{\hbar\omega_{rf}}{\mu_B |g_F|} \right)^2 + \left(\frac{\mathbf{B}_{rf\perp}(\mathbf{r})}{2} \right)^2 \right]^{1/2} . \end{aligned} \tag{2.25}$$

The first term inside the square root is called resonance term. Its sole dependence on the driving field comes from the RF frequency ω_{rf}. Furthermore, it is spatially dependent due to the inhomogeneity of the static field and vanishes at points with zero detuning Δ. The coupling term, the second part of the square root, depends on the absolute effective magnetic field component, which is perpendicular to the static field. Because both the static and the driving field are inhomogeneous, the spatial dependence of the effective field can be rather complex.

2.3.3 Numeric calculation of dressed potentials

For a numeric calculation of dressed potentials, some steps from above can be streamlined to make the code simpler and shorter. Instead of solving the full Hamiltonian of equation 2.13, it is only necessary to use the potential energy term

$$H_{initial} = g_F \mu_B \mathbf{B}(\mathbf{r}, t) \cdot \mathbf{F}. \tag{2.26}$$

The total magnetic field $\mathbf{B}$ consists of a static magnetic field $\mathbf{B}_s$ and an RF term $\mathbf{B}_{rf}$. Instead of a complex driving field $\mathbf{B}_{rf}$, the approach for the oscillating field is

$$\mathbf{B}_{rf}(\mathbf{r}) = \mathbf{B}^A_{rf}(\mathbf{r}) cos(\omega_{rf} t) + \mathbf{B}^B_{rf}(\mathbf{r}) cos(\omega_{rf} t + \gamma). \tag{2.27}$$

Similar to section 2.3.1, the unitary transformation U_S is applied to the Hamiltonian. The system can be transferred into a frame rotating with the angular frequency ω_{rf} around the axis of the local static field by applying another unitary transformation $U_R = exp[-i\frac{g_F}{|g_F|} F_z \omega_{rf} t]$. In the rotating wave approximation (RWA, see section 2.3.2), the terms that rotate with frequency $2\omega_{rf}$ can be neglected. The result is a time-independent Hamiltonian

$$H = \left[g_F \mu_B |\mathbf{B}_s(\mathbf{r})| - \frac{g_F}{|g_F|} \hbar\omega_{rf} \right] F_z + \frac{g_F \mu_B}{2} \begin{pmatrix} \bar{B}_x \\ \bar{B}_y \end{pmatrix}^T \begin{pmatrix} F_x \\ F_y \end{pmatrix} . \tag{2.28}$$

The rotated RF field $\bar{\mathbf{B}}$ is described by

$$\bar{\mathbf{B}} = R_S \mathbf{B}^A_{rf}(\mathbf{r}) + R_\delta R_S \mathbf{B}^B_{rf}(\mathbf{r}), \tag{2.29}$$

where the rotation matrix R_S performs the unitary transformation U_S, given by equation 2.6. The rotation matrix R_δ originates from U_R and takes the phase shift of the second RF field component (with respect to the first one) into account. The angle δ is given by

$$\delta = -\frac{g_F}{|g_F|} \gamma. \tag{2.30}$$

Please note that the sign of δ depends on the sign of g_F, so different hyperfine states see different potentials. After applying those rotations to the RF field, one can calculate the potential energy of the respective dressed state with

$$V_{ad}(\mathbf{r}) = g_F \mu_B \tilde{m}_F \sqrt{\left[|\mathbf{B}_s(\mathbf{r})| - \frac{\hbar\omega_{rf}}{|g_F|\mu_B}\right]^2 + \frac{1}{4}\left[\bar{B}_x^2 + \bar{B}_y^2\right]}. \tag{2.31}$$

This calculation uses the RWA, which has to be checked for validity in the trapping area. [11]

2.4 Atom chips

In the experimental setup atom chips are used to produce magnetic traps. Atom chips are small chips with side lengths of a few mm and a thickness of a few µm. Various magnetic field shapes can be realized using wire structures on the chip surface.

2.4.1 Advantages

In conventional systems, large coils or permanent magnets produce the trapping field. For a sufficiently large field gradient for atom traps and evaporative cooling, currents up to a few hundred amperes have to be used. A portable approach to trapping atoms without those huge currents is the use of atom chips.

The Biot-Savart law for the magnetic field of a wire is given by

$$B = \frac{\mu_0 I}{2\pi r} \propto \frac{I}{r}, \tag{2.32}$$

with the gradient

$$B' = -\frac{\mu_0 I}{\pi r^2} \propto \frac{I}{r^2}. \tag{2.33}$$

Therefore, the gradient increases with bigger currents and smaller distance to the wire. Since the dependency on the distance is quadratic, the atoms should be placed as close as possible to the wire in order to achieve a large gradient. The gas of atoms is usually confined in a vacuum chamber, which limits the possible distance to the field-generating structure if coils are used instead of a chip. Consequently, another advantage of atom chips in comparison to coils is the cooling of the wires: As due to the proximity, lower current is needed in each wire, less cooling is necessary. More information regarding atom chips can be found in [12].

2.4.2 Chip traps

The content of the following subsection is mainly summarized from [13], where more details regarding atom chip traps and guides can be found.

One-wire guides

By superimposing the field of a current-carrying wire mounted on a chip surface with a homogeneous external bias field, a magnetic guide can be generated. If the bias field is parallel to the chip surface and perpendicular to the wire, the guide is called *side guide*, and atoms in a weak-field-seeking state are guided in a potential tube along a line parallel to the wire. If the angle between the wire and the bias field varies, the longitudinal field component with respect to the guide and the potential minimum changes, too. This results in a trap or barrier for the guided atoms and impedes guiding a BEC.

On the other hand, if the bias field is perpendicular to the chip surface (i.e., oriented along the vertical axis), the potential minimum is independent of the wire direction and moves into the plane of the chip.

Two-wire guides

Using two wires with antiparallel currents instead of one and a vertical bias field to form a magnetic guide yields the advantage, that the wires can be bent in any way on the chip surface without creating a barrier. The height of the guide above the surface depends on the bias field strength B_b relative to the critical bias field

$$B_{crit} = \frac{\mu_0 I_w}{\pi d}, \tag{2.34}$$

with the vacuum permeability μ_0, half the distance between the wires d, and the wire current I_w.

If $B_b > B_{crit}$, both wires form separate side guides (figure 2.8 (d)), which are located between the wires on the chip surface.

For $B_b = B_{crit}$, a single guide is formed between the wires, which is still located in the plane of the chip. (figure 2.8 (c)).

For $B_b < B_{crit}$, two minima, one above the surface and the other below it, are created (figure 2.8 (a) and (b)) at distance r_0 to the chip surface:

$$r_0 = d\sqrt{\left(\frac{\mu_0}{\pi}\right)\frac{I_w}{dB_b} - 1} \tag{2.35}$$

The magnetic field close to the minimum is a 2D quadrupole field with gradient

$$\left.\frac{\mathrm{d}B}{\mathrm{d}r}\right|_{r_0} = \left(\frac{2\pi}{\mu_0}\right)\frac{B_b^2}{I_w}\frac{r_0}{d}. \tag{2.36}$$

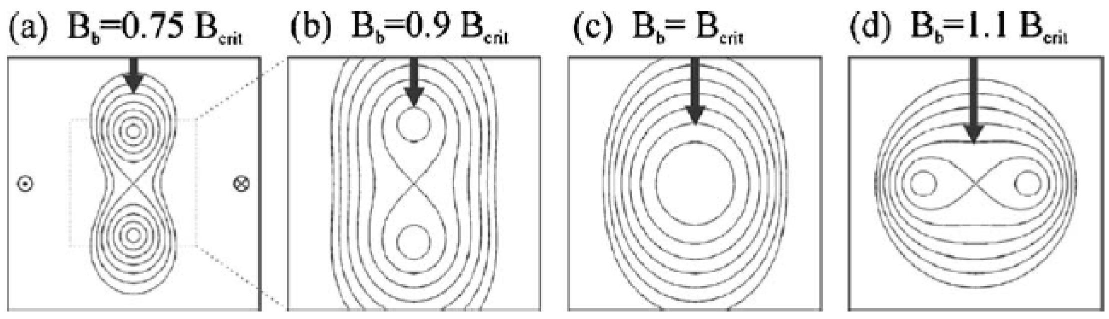


Figure 2.8: Equipotential lines for the magnetic potentials in a two-wire guide with antiparallel currents and a vertical bias field [13].

Traps

Three-dimensional confinement can be reached by adding another two wires to the already existing setup of two parallel ones. They are also called "end caps" since they close the trap.

A quadrupole-like trap can be reached by running antiparallel current through the end caps as well. A picture of this design is shown in figure 2.9 (a). Figure 2.10 depicts the respective potentials for different bias field strengths and for a distance $d = 57.5\,\mu m$ between the two initial wires, distance $D = 180\,\mu m$ between the end caps and currents of 2 A. In this case, the overall magnetic field of the two wires and the end caps creates a trap by itself, but the use of a bias field moves the trap towards the chip surface. Figure 2.9 (b) shows a different design with similar effects but without the need of crossing wires.

On the other hand, if the currents in the end caps are parallel to each other, the trap minimum gets lifted and the form of the potential is changed from linear to harmonic. This yields a Ioffe-Pritchard-like trap. The setup is shown in figure 2.9 (c), the equipotential lines for currents of 2 A and varying bias field strengths are shown in figure 2.11. In analogy to the quadrupole-like trap, figure 2.9 (d) shows a different setup with, again, similar effects, but without needing to cross wires.

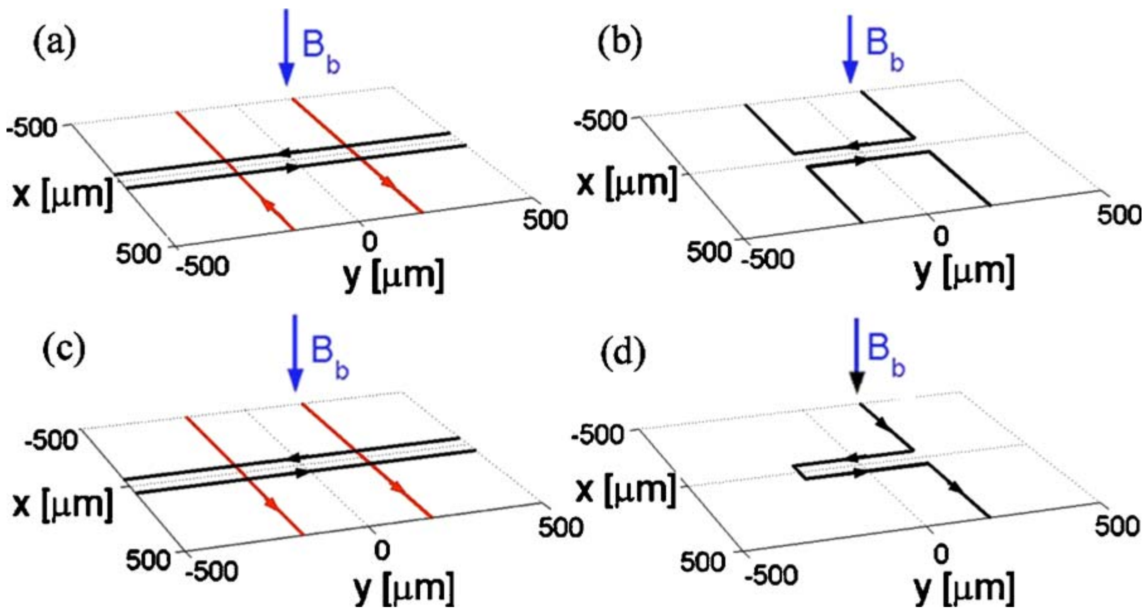


Figure 2.9: Chip designs for forming a magnetic trap [13].

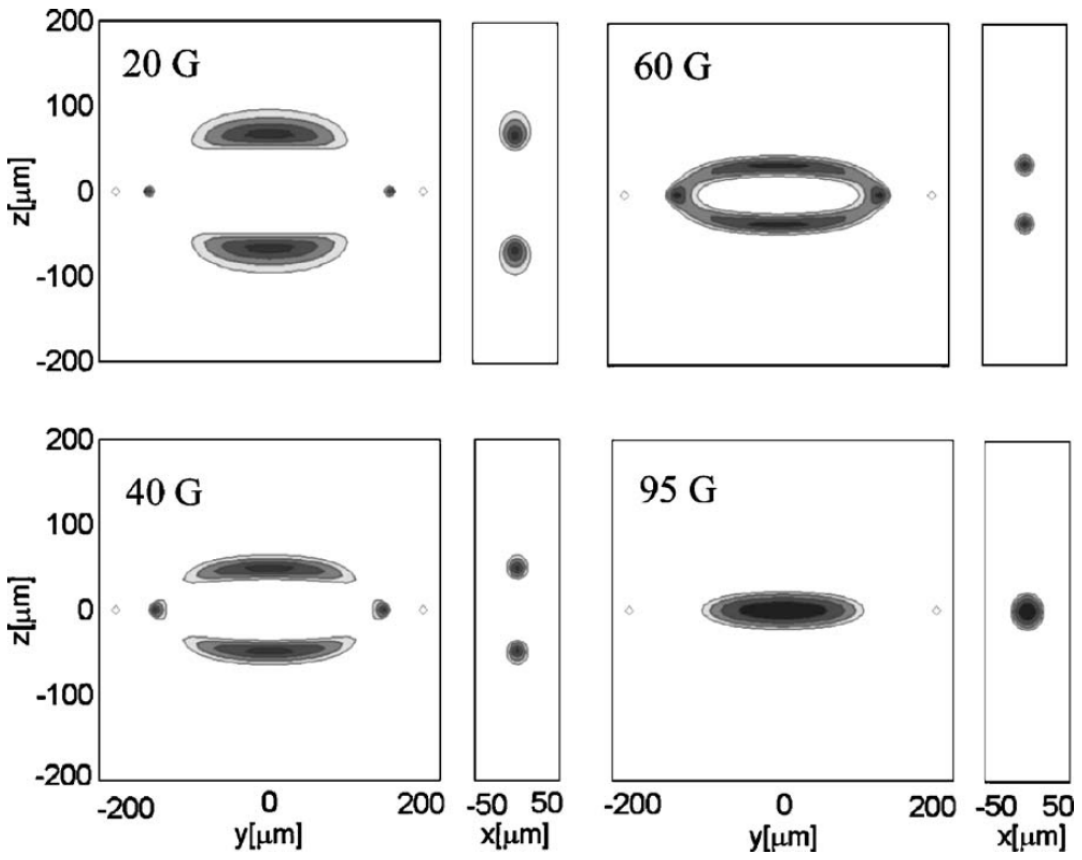


Figure 2.10: Equipotential lines of the magnetic potentials in a two-wire vertical quadrupole-like trap [13].

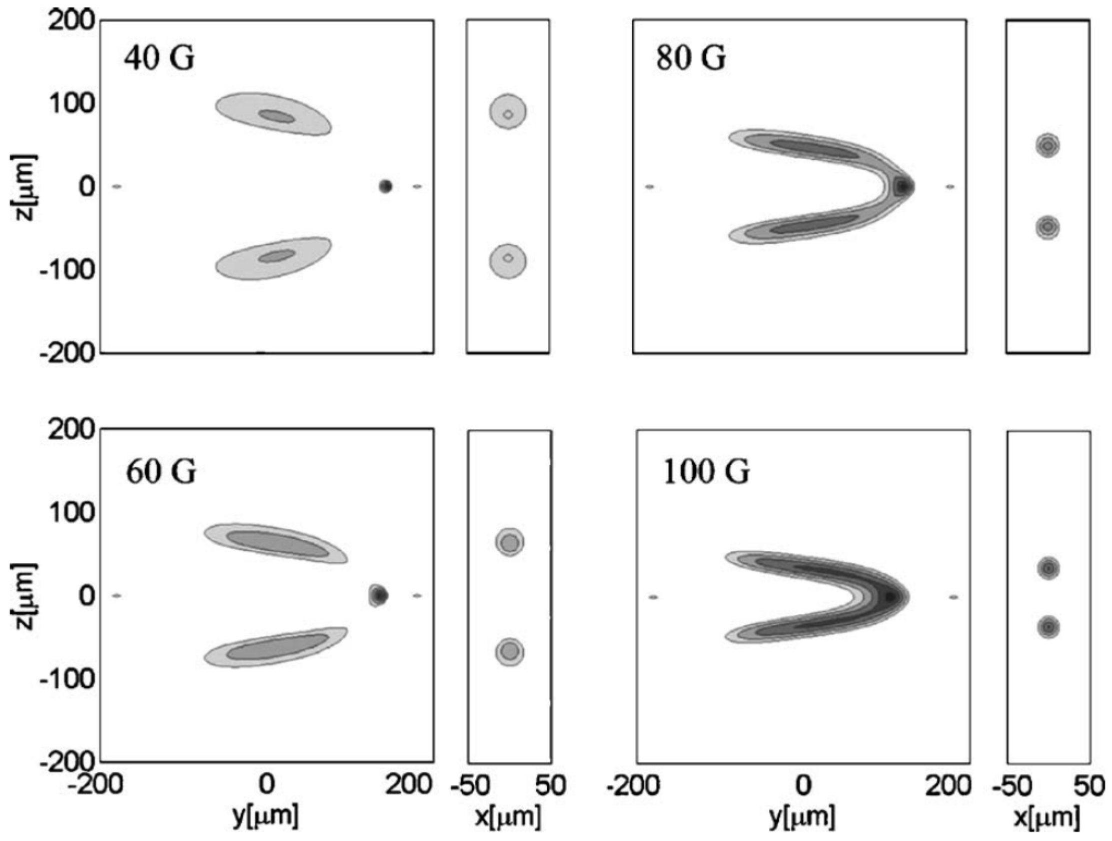


Figure 2.11: Equipotential lines of the magnetic potentials in a two-wire vertical Ioffe-Pritchard-like trap [13].

3 Realization

3.1 Experimental realization

The centerpiece of the experiment is a vacuum glass cell located in the middle of four coil pairs and beneath an atom chip. In a typical experimental cycle, Caesium atoms are transferred to and then captured in the glass cell. The setup is shown in figure 3.1, the cell in figure 3.2.

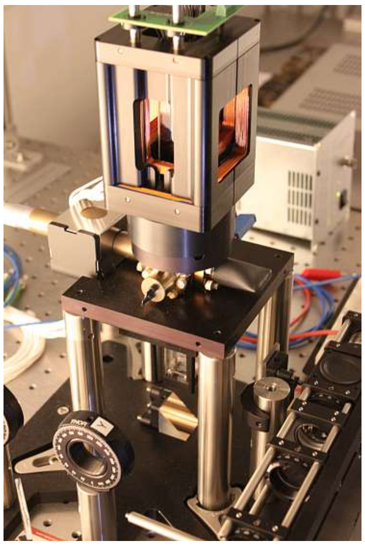


Figure 3.1: Photo of the whole setup including glass cell, coil cube and atom chip.

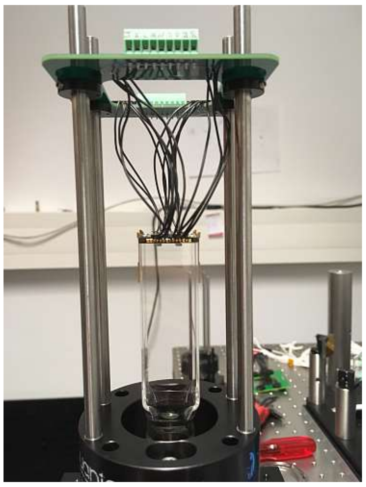


Figure 3.2: Photo of the vacuum glass cell.

3.1.1 Coil cube

The four coil pairs surrounding the glass cell are placed in a construction called coil cube. It is shown together with the coil labels in figure 3.3. Most of them are used for loading the Caesium atoms into the trap. The only one of relevance for this thesis is the Z1/Z2 coil pair, which forms the magnetic traps together with the atom chip. The coordinate system of the coil cube is shown in table 3.1. The field directions for positive technical current are given by table 3.2.

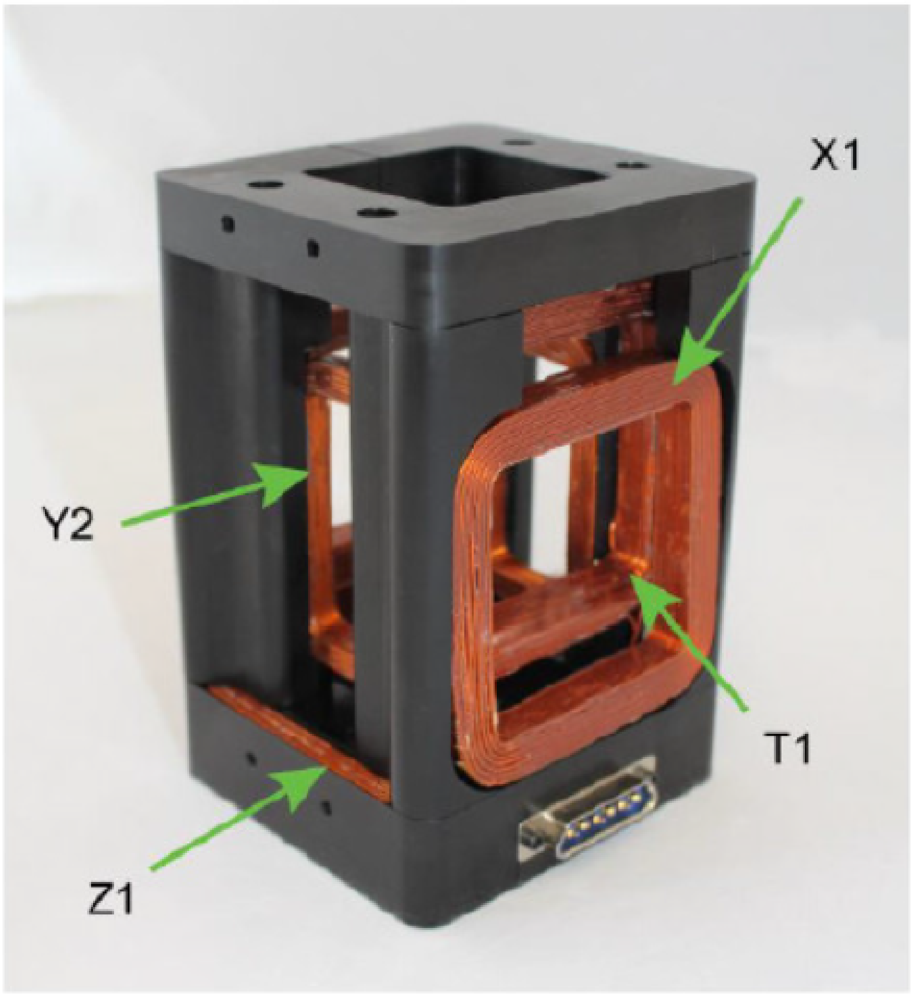


Figure 3.3: Coil cube around the glass cell with labels.

Axis	direction
X	X1 → X2
Y	Y2 → Y1
Z	Z2 → Z1

Table 3.1: Orientation of the coordinate axes relative to the coils. For coil numbering cf. figure 3.3.

Coil	current orientation	field direction
X1	anti-clockwise	-X (outwards = right)
X2	anti-clockwise	-X (inwards = right)
T1	anti-clockwise	-X (outwards = right)
T2	anti-clockwise	-X (inwards = right)
Y1	clockwise	+Y (outwards = right)
Y2	clockwise	+Y (inwards = right)
Z1	clockwise	+Z (outwards = down)
Z2	clockwise	+Z (inwards = down)

Table 3.2: Current and field directions of the coils for positive technical current. Spatial orientations are given for front view in figure 3.3, axial directions w.r.t. table 3.1.

3.1.2 Atom chip

The atom chip was manufactured by ColdQuanta and is shown in figure 3.4. All information is taken from the delivery sheet [14]. The chip is placed as the top boundary of the vacuum glass cell. The chip consists of inner and outer trap wires, which are parallel to each other, and perpendicular H wires. The first ones are placed on the chip bottom, so they are located directly in the glass cell. The H wires, on the other hand, are located on top of the chip. Some size and material properties of the atom chip are shown in table 3.3.

Chip material parameters	
Substrate thickness	420 µm
Vacuum side metallization	Brightened copper
Ambient side metallization	Gold
Metallization thickness	10 µm
Dielectric layer	∼0.4 µm SiO_2

Table 3.3: Atom chip size and material parameters [14].

A schematic of the atom chip used in this experiment is shown in figure 3.5. Its wire connections and their labels are given by table 3.4, the coordinate system with respect to the schematic can be found in table 3.5. In the lab coordinate system, the bottom chip surface lies in the plane where the vertical component z is zero. Therefore, the vertical centers of the trap wires and the H wires are located at z = +5 µm and z = -425 µm, respectively. All wires are 10 µm thick (cf. table 3.3), which results in an additional ± 5 µm when considering the wire center.

Wire	+	-	shorted	field direction
Inner Trap	A	R	I & J	-X: -Z (up) / +X: -Z (up)
Outer Trap	B	Q	H & K	-X: -Z (up) / +X: -Z (up)
Inner H	F	D	M & O	-Y: -Z (up) / +Y: -Z (up)
Outer H	G	C	L & P	-Y: -Z (up) / +Y: -Z (up)

Table 3.4: Chip wire connections according to the schematic in figure 3.5. Field directions are given for positive technical current.

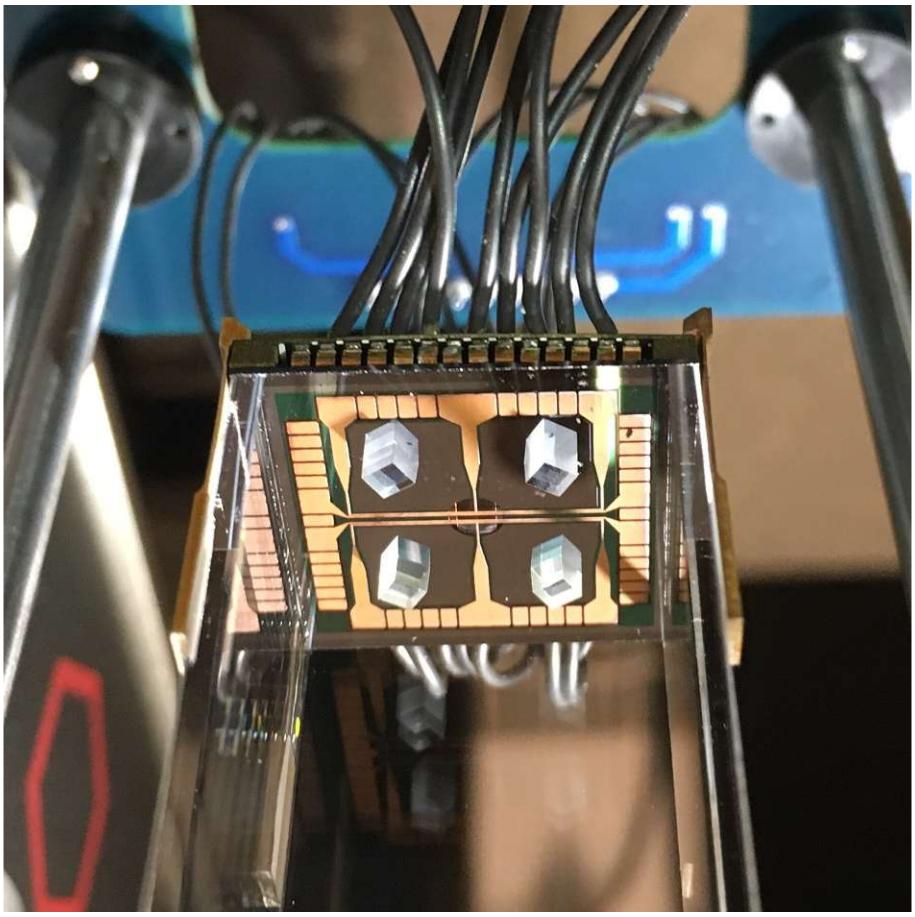


Figure 3.4: Photo of the bottom of the atom chip.

Axis	**direction**
X	$\downarrow$
Y	$\leftarrow$
Z	$\otimes$

Table 3.5: Orientation of the coordinate axes in figure 3.5.

3.1.3 Atom chip traps

By comparing the design of the atom chip in figure 3.5 and the wire setups that form a trap in figure 2.9, it can be concluded, that the currents in the trap wires need to be antiparallel. A quadrupole trap requires antiparallel H wire currents as well, while a Ioffe-Pritchard-like trap demands parallel currents in the H wires. According to the coordinate system, wire connections and field directions, the currents in the trap wires, H wires and the Z coils need to all carry the same sign to form a trap. One possible combination of wires is antiparallel -*2 A* in the inner trap wires, (anti-) parallel -*3 A* in the outer H wires and -*2 A* in the Z coils, forming a Ioffe-Pritchard (quadrupole) trap.

3.1.4 Waveform generator

The RF field has to be generated by a dual-channel waveform generator. In this experiment a Tabor WW5062 is used. Details on its control software can be found

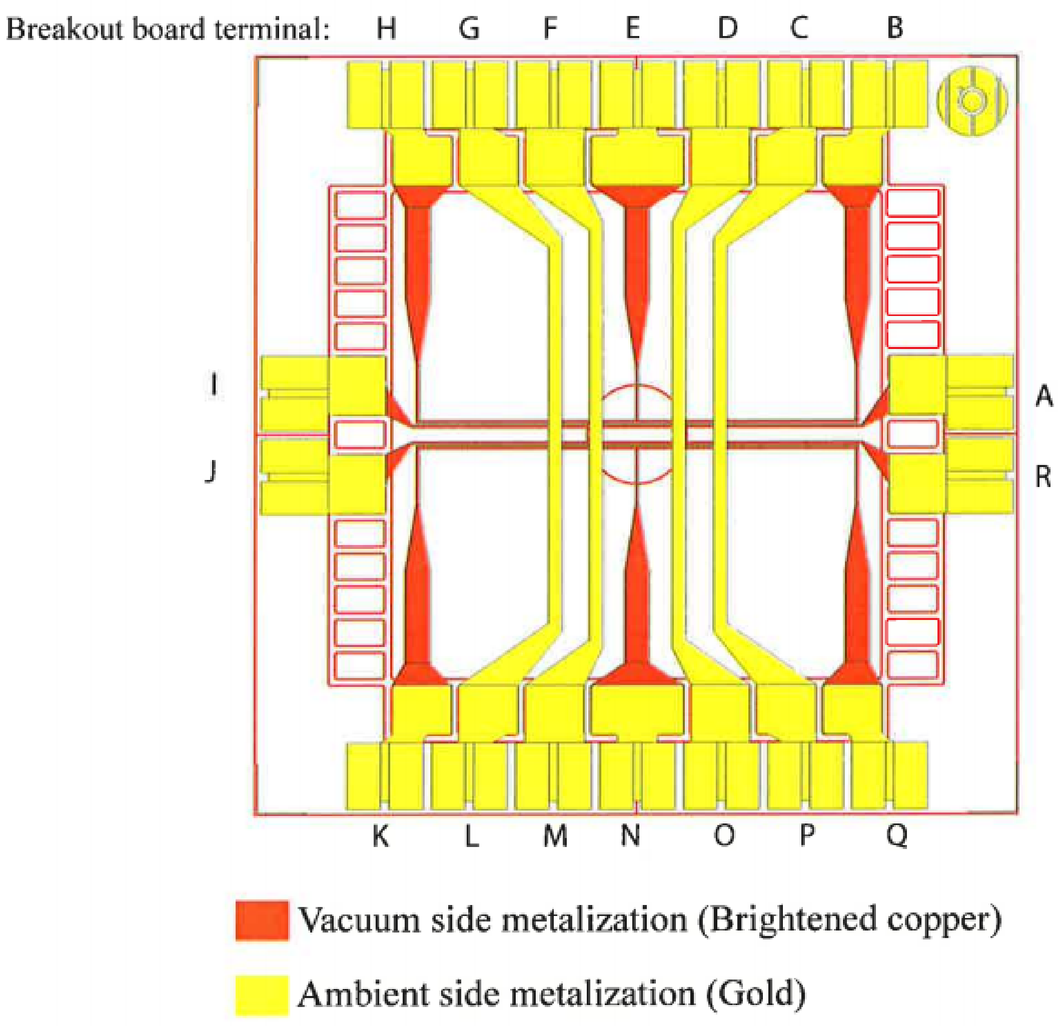

Breakout board terminal:
H
G
F
E
D
C
B
I
J
A
R
K
L
M
N
O
P
Q
🟥 Vacuum side metalization (Brightened copper)
🟨 Ambient side metalization (Gold)

Figure 3.5: Schematic of the atom chip [14]

later in section 3.4. It has two output channels that can provide different amplitudes and phases. The frequency is identical for both channels. One of its standard waveforms is a sine to which the subsequent calculations will be limited. The frequency range covers 100 µHz up to 25 MHz. The phase can be adjusted from 0° to 360° with a resolution of 0.1°.

It also has an analog amplitude modulation (AM) input, which multiplies a factor between 0 and 1 to the amplitude. It accepts input voltages between 0 V and +5 Vp-p. The factor ranges from 1 for 0 V to 0 for 5 V applied voltage.

More details on the Tabor WW5062 can be found in the associated datasheet [15].

3.2 Approximations

Some approximations were necessary to calculate the potential energy of dressed atoms numerically.

3.2.1 Biot-Savart law for non-static currents

The Biot-Savart law in equation 2.32 is used for calculating the magnetic field of current-carrying wires. It usually only applies to static currents. However, if the periodic change in current is small compared to the time t the field needs to get from the wire to any relevant point in space, one can still use the Biot-Savart law for oscillating currents. In that case, the time-dependence of the current is transferred to the magnetic field.

For the approximation, the distance from any H wire (see section 3.1.2) to the lower end of the atom cloud right below the wire was used. With the actual chip surface being at $z = 0\ \mu m$, the height of the H wires is $h_H = -425\ \mu m$ according to the lab coordinate system introduced in section 3.1.2. Furthermore, the lower end of the atom cloud is approximately at $h_{cloud} = 400\ \mu m$. The overall distance that the field has to travel from an H wire to the end of the atom cloud thus is $\Delta h = h_{cloud} - h_H = 825\ \mu m = 8.25 \cdot 10^{-4}\ m$. The magnetic field travels with the speed of light, which can be approximated by $v = c = 3 \cdot 10^8\ \frac{m}{s}$. Consequently, the time t in which the field reaches the bottom end of the atom cloud would be $t = \frac{\Delta h}{v} = 2.67 \cdot 10^{-12}\ s$. Taking a usual RF frequency of $f = 100\ MHz$ yields 10^8 oscillations per second. In $t = 2.67 \cdot 10^{-12} s$, only $n = f \cdot t = 10^8 s{-}^1 \cdot 2.67 \cdot 10^{-12} s = 2.67 \cdot 10^{-4}$ oscillations take place. The oscillation period of the RF field thus is much shorter than the travel time of the field from the wire to the atom cloud, and it is appropriate to use the Biot-Savart law for RF currents as well.

3.2.2 Spatial independence of the phase shift

The phase shift between the currents of two parallel wires does not automatically transfer to the magnetic field. For each spatial position, the correct phase shift depends on the path difference of both magnetic fields. Since the deviation from the initial phase shift is below 1% everywhere inside the volume of interest, the phase shift is approximated to be independent of the position.

3.3 Numerical potential calculation

This chapter covers the realization of the theory in section 2.3.3.

The potential energy of dressed Caesium atoms is calculated numerically using MATLAB version R2020a. The computation of a static trap with the experimental setup described in section 3.1 was done before by Clemens Maderböck [16] and later modified by the members of the research group. That work was the starting position for the calculation of the dressed states.

For version control, a remote GitHub repository is used.

3.3.1 First steps

The first step for calculating the potential energy is applying the Biot-Savart law (equation 2.32). In order to enhance efficiency and runtime, the magnetic field is calculated as normalized field, short *normfield*: For each wire and coil, the magnetic field is calculated for a current of 1 A. The result is a 4D array assigning a magnetic field vector to each spatial point *(x,y,z)*. Multiplying each normfield with the desired current value later in the code yields the final magnetic fields. Since the creation of normfields is time-consuming, a uniform grid was chosen and the associated normfields were determined and saved. They now only have to be loaded into the MATLAB workspace using the script *Load_ Grid_ Normfields* before starting the computation. The numeric potential calculation needs the following inputs in order to function properly:

- x, y, z: 1D arrays defining the grid size and containing the coordinates associated with the grid points
- normfields: 4D arrays representing the normalized magnetic fields for the different wires/coils
- available_elements: 1D array containing indices representing every used wire/-coil

The grid used for this thesis is shown in table 3.6.

	Range Start [mm]	**Range End [mm]**	**# Grid Points**
x	-0.15	+0.15	61
y	-2	+2	801
z	0	0.6	121

Table 3.6: Grid parameters used in the calculations.

3.3.2 User input

After loading the grid and normfields, the user has to configure the calculation. The parameters set by the user can be split into three groups:

- General parameters (see listing 3.1): number of simulation steps, data precision, contributions to the potential energy, atomic state
- Static field parameters (see listing 3.2): currents for each pair of wires and coils, variables for pairwise or separate use of the chip wires
- RF field parameters (see listing 3.3): currents for each wire, general RF frequency for all wires, phase shift for each wire, RWA check settings (see more details about the RWA check parameters in section 3.3.3)

```
%%% Choose number of steps for sequence simulation and precision %%%
% A sequence simulation can be written by defining e.g. currents in the
% form linspace(I_i,I_f,number_of_steps) or by writing a custom array of
% the correct length
number_of_steps = 1;
precision = 'double';

%%% Choose which types of potentials should be considered in the
    calculations %%%
% Directly assigned to structure which is passed to the functions
addpot.static_CeBECi = 1;
addpot.static_Coils = 0;
addpot.RF = 1;
addpot.MW = 0;
addpot.dipole = 0;
addpot.gravity = 1;

%%% Choose atomic state %%%
atom_state = [4,4]; % array of the form [F,m_F]
```

Listing 3.1: General user input parameters.

```
%%% Set currents in Amps %%%
% Set a current to NaN to completely ignore the associated coil/wire in the
    calculations
MOT_curr = NaN;
Transf_curr = NaN;
Y_curr = NaN;
Z_curr = -2;
Chip_InnerTrap1_curr = -2;
Chip_OuterTrap1_curr = NaN;
Chip_InnerH1_curr = NaN;
Chip_OuterH1_curr = -3;
% Subsequent currents only taken into account if wires are selected for
    separate use below
Chip_InnerTrap2_curr = NaN;
Chip_OuterTrap2_curr = NaN;
Chip_InnerH2_curr = NaN;
Chip_OuterH2_curr = NaN;

%%% Select pairwise or separate use of chip wires %%%
% 0 for separate, 1 for parallel, -1 for antiparallel currents
Chip_InnerTrap_pair = -1;
Chip_OuterTrap_pair = -1;
Chip_InnerH_pair = -1;
Chip_OuterH_pair = 1;
```

Listing 3.2: Static field user input parameters.

```
% RF current amplitudes in A (steps possible)
InnerTrap1_amp_RF = NaN;
InnerTrap2_amp_RF = NaN;
OuterTrap1_amp_RF = -0.5;
OuterTrap2_amp_RF = 0.5;
InnerH1_amp_RF    = NaN;
InnerH2_amp_RF    = NaN;
OuterH1_amp_RF    = NaN;
OuterH2_amp_RF    = NaN;

% RF frequency in Hz (normal frequency) (steps possible)
frequency_RF = 1e5;

% RF phases in rad (steps not allowed)
```

```
InnerTrap1_phase_RF = NaN;
InnerTrap2_phase_RF = NaN;
OuterTrap1_phase_RF = 0;
OuterTrap2_phase_RF = -pi/2;
InnerH1_phase_RF    = NaN;
InnerH2_phase_RF    = NaN;
OuterH1_phase_RF    = NaN;
OuterH2_phase_RF    = NaN;

% If check_RWA_RF equals 1, the RWA checks will be executed inside of
    effective_RF_field.
% The output is visual. Every red dot stands for a point where the
    respective condition is not satisfied.
% check_RWA_RF_stepsize stands for the size of areas the whole grid will be
    split into to make the output more understandable.
% So for example, if the stepsize equals 5 then every point in the output of
     the RWA checks will represent a 5x5x5 point area of the original grid (
    besides the edges of course).
% The results of the RWA checks in each area will be a disjunction of each
    single point:
% If the condition fails at any point inside an area then the whole area did
     not satisfy the condition.
% An index comparison of the new and original grid is written to
    indexfile_check_RWA_RF.xlsx.
check_RWA_RF = 1;
check_RWA_RF_stepsize = 5;
```

Listing 3.3: RF field user input parameters.

The format of some of the user input parameters depends on the simulation type. A single-step simulation requires all the currents, frequencies and phases to be of type *double*. For a sequence simulation, all of the relevant parameters have to be arrays of length *number_ of_ steps*. The parameters that can be varied in a sequence are the currents for each wire or coil (static and RF) and the RF frequency. In order to make the program more user-friendly, the tool can deal with deviations from the necessary variable type. The full method is shown as a flowchart in figure 3.6.

3.3.3 Calculation of the magnetic field and the adiabatic potential

Preparation of the user input for the calculation

After configuring and starting the simulation, the tool needs to prepare the inputs for the further procedure. First, the currents, frequency and phases are checked for the correct dimension (cf. figure 3.6) and written to arrays. The responsible functions are *write_ current_ sequence_ CeBECi* for the static and *write_ RF_ parameter_ sequences* for the RF parameters. Details on their input and output values are shown in listings 3.4 and 3.5.

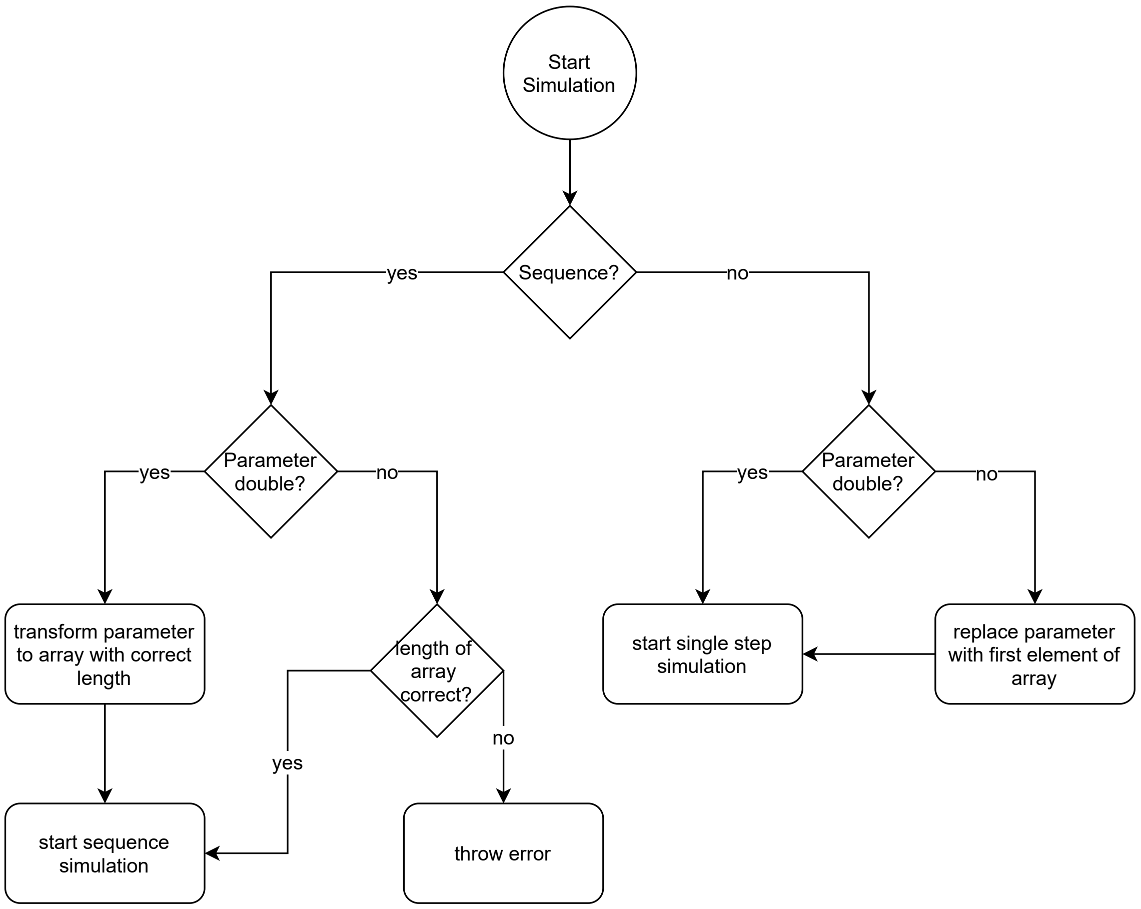

Start Simulation
Sequence?
yes
no
Parameter double?
yes
no
Parameter double?
yes
no
transform parameter to array with correct length
length of array correct?
start single step simulation
replace parameter with first element of array
yes
no
start sequence simulation
throw error

Figure 3.6: Dealing with different variable and simulation types. Possible variable types are limited to *double* and *array*.

```
function currents = write_current_sequence_CeBECi(execute_function,
   number_of_steps,currents_cell,wire_pairs_arr,available_elements)
% Checks currents of static field for the correct dimension and
% writes them into a 2D array.

% Parameters
% currents           [out]: 2D array with currents for each wire/coil
% execute_function   [in] : Boolean determining whether this function has
%                           to be executed
% number_of_steps    [in] : Integer determining the simulation steps
% currents_cell      [in] : Cell array containing user input current for
%                           each wire/coil
% wire_pairs_arr     [in] : Array determining pairwise/separate use for
%                           each wire/coil pair
% available_elements [in] : Array determining every wire/coil that is
%                           used for the simulation
```

Listing 3.4: Function for preparing static field current inputs.

```
function [currents_RF,frequency_RF,phases_RF] = write_RF_parameter_sequences
    (execute_function,number_of_steps,cell_currents_RF,frequency_RF,
    cell_phases_RF,available_elements)
% Checks currents, frequency and phases of RF field for the correct
% dimension and writes them into arrays.

% Parameters
% currents_RF        [out]: 2D array with currents for each wire
% frequency_RF       [out]: 1D array with frequency for each simulation
%                           step
% phases_RF          [out]: 1D array with phases for each wire
% execute_function   [in] : Boolean determining whether this function has
%                           to be executed
% number_of_steps    [in] : Integer determining the simulation steps
% cell_currents_RF   [in] : Cell array containing user input current for
%                           each wire
% frequency_RF       [in] : Double or 1D array with RF frequency
% cell_phases_RF     [in] : Cell array containing user input phase for
%                           each wire
% available_elements [in] : Array determining every wire/coil that is
%                           used for the simulation
```

Listing 3.5: Function for preparing RF field parameter inputs.

Next, the tool has to ensure that every used variable exists in the MATLAB workspace. The functions called later rely on those parameters and raise an error if any of them don't exist. The script *Prepare_MagneticFieldInputs* deals with this issue and checks the variables for reasonable dimensions and values.

Calculation of the effective magnetic field

After preparing the data, the tool can proceed to calculate the effective magnetic field using the function *calculate_magnetic_field*. All the other functions mentioned in this paragraph, whether it be for calculating the static field, the RF field amplitudes or the effective field, are invoked inside of *calculate_magnetic_field*. A brief parameter documentation is shown in listing 3.6.

The static magnetic field of each wire or coil is calculated by multiplying the respective current to the associated normalized field. Eventually, the different contributions to the total static field are added up and a 4D array assigning a field vector to each grid point is returned. All of this is done by the function *static_field*, whose parameters are explained in listing 3.7.

Analogous, the product of current and normalized field yields the RF field amplitude of each chip wire. Equation 2.27 shows that the field amplitudes can be interpreted as static fields as well, the oscillatory part being given by the sine. Section 3.2.1 also shows that the use of the Biot-Savart law is valid for radio-frequency applications with small distances between atoms and wires. The function *rf_field*, whose parameters are shown in listing 3.8, performs the calculation with the same normfields as the function *static_field*, since the same wires are used for the RF signal, but without the sum over the contributions because of the phase difference.

```
function [B_static_vec,B_eff_abs] = calculate_magnetic_field(addpot,
    precision,x,y,z,number_of_steps,normfields_CeBECi,currents_CeBECi,
    normfields_Coils,currents_Coils,atom_state,currents_RF,frequency_RF,
    phases_RF,check_RWA_RF_properties)
% Calculate the effective magnetic field from each contribution

% Parameters
% B_static_vec         [out]: 4D array determining the vectorial static
%                             field
% B_eff_abs            [out]: 3D array determining the absolute effective
%                             field
% addpot               [in] : Struct determining the different
%                             contributions to the magnetic field
% precision            [in] : String determining the precision of the
%                             data
% x                    [in] : 1D array determining the grid points on the
%                             x-axis
% y                    [in] : 1D array determining the grid points on the
%                             y-axis
% z                    [in] : 1D array determining the grid points on the
%                             z-axis
% number_of_steps      [in] : Integer determining the simulation steps
% normfields_CeBECi    [in] : 1D cell array determining the normfields
%                             for each wire/coil
% currents_CeBECi      [in] : 2D array determining the currents for the
%                             static field
% normfields_Coils     [in] : 1D cell array determining the normfields
%                             for levitation and feshbach coils
% currents_Coils       [in] : 2D array determining the currents for the
%                             levitation and feschbach coils
% atom_state           [in] : 1D array determining the atom state [F,m_F]
% currents_RF          [in] : 2D array determining the currents for the
%                             RF field
% frequency_RF         [in] : Double/1D array determining the frequency
%                             of the RF field
% phases_RF            [in] : 1D array determining the phases of the RF
%                             field
% check_RWA_properties [in] : 1D array determining the check_RWA
%                             properties
```

Listing 3.6: Function for calculating the effective magnetic field.

```
function static_field_vec = static_field(normfields,currents)
% Multiply the current to the respective normfield and sum over all
% contributions to the static field.

% Parameters
% static_field_vec [out]: 4D array determining the vectorial static
%                         magnetic field
% normfields       [in] : 1D cell array determining the normfields for
%                         each wire/coil
% currents          [in] : 1D array determining the currents for each
%                          wire/coil
```

Listing 3.7: Function for calculating the static magnetic field from normalized fields.

```
function rf_fields = rf_field(normfields_CeBECi,currents_RF)
% Multiply the RF currents to the CeBECi normfields (no sum over wires yet
   because of different phase shifts)

% Parameters
% rf_fields         [out]: 1D cell array containing a 4D array for each
%                          wire. The 4D array is the magnetic field
%                          amplitude vector of the respective
%                          contribution to the RF field.
% normfields_CeBECi [in] : 1D array determining the normfields for each
%                          wire/coil
% currents_RF       [in] : 1D array determining the currents for each
%                          wire
```

Listing 3.8: Function for calculating the RF magnetic field from normalized fields.

All the values necessary for calculating the effective magnetic field in equation 2.21 are available now, and the function *effective_RF_field* is called. All input and return parameters are listed and briefly explained in listing 3.9. The calculation consists of four major steps:

1. Calculation of the rotation angles α (equation 2.3) and β (equation 2.4):

   The angles α and β depend on the orientation of the static magnetic field and thus on the spatial position or grid point. Therefore, they are calculated as a 3D array to match the grid coordinates. In MATLAB, it is possible to modify only array elements fulfilling a specific condition, which makes it easy to implement the additional subtraction of 180° for the relevant grid points. This step is shown in listing 3.10.

2. Calculation of the rotated magnetic field $\bar{\mathbf{B}}$ (equation 2.29):

   Due to different phase shifts, the individual terms of $\bar{\mathbf{B}}$ have to be calculated separately. Therefore, the first *for* loop iterates over the chip wires. The rotation angle δ is given for each contribution by equation 2.30. Since the rotation matrices have to be applied to the whole magnetic field or, in programming terms, to each grid point separately, there are three more *for* loops necessary, one for each axis. Every iteration step of the nested loop represents one grid point and, consequently, one magnetic field vector. The rotation matrices R_S and R_δ are now applied to each local magnetic field vector. R_S can be split into a rotation of α around the z-axis and a subsequent rotation of β around the y-axis thus be defined as $R_y(\beta)R_z(\alpha)$. The rotation R_δ usually rotates any vector around the direction vector of the local static magnetic field. After applying R_S, the local static field vector is aligned with the z-axis. As a consequence, R_δ rotates around the z-axis and can therefore be written as $R_z(\delta)$. After taking the phase shift into account, the contributions of each wire to the rotated RF field can finally be added up to form a single vectorial RF field.

The loops and the application of rotation matrices to each RF field vector are shown in listing 3.11.

3. Calculation of the absolute effective RF field

   The final step is the calculation of the absolute effective RF field. This translates to the square root in equation 2.31 and is shown in listing 3.12. The result is a 3D array assigning an absolute effective magnetic field value to each grid point.

4. Checking the RWA conditions

   The rotating wave approximation consists of three separate parts described in section 2.3.2. The two most important conditions can be summarized to

$$\Omega, \Delta \ll \omega_0, \tag{3.1}$$

with the Rabi frequency Ω, the detuning Δ and the Larmor frequency ω_0 [8]. The third condition can usually be neglected for the simulation. Nevertheless, the function *check_RWA_conditions* calculates all three relations in section 2.3.2 and additionally combines them into one result using the logical *OR* operator.

It was shown that the RWA only breaks down for RF field amplitudes that are larger than the static field amplitudes [17]. Normally, this doesn't apply to the experimental realization or the user input, which is why the operator $<$ was used in all calculations instead of $\ll$.

The RWA condition checks can be activated by setting the variable *check_RWA_RF* to *1*. In that case, the function *check_RWA_conditions* is called at the end of *effective_RF_field*. A brief parameter documentation can be found in listing 3.13. The results are displayed visually in the form of 3D plots, where each grid point, whose magnetic field doesn't satisfy the respective condition, is marked red. Because this method can get quite unclear for $61 \cdot 801 \cdot 121 = 5912181$ grid points, the whole grid can be coarsened. A new grid is introduced, where every point represents a cubic volume of the initial pattern – except the edges of course. The side length of that volume is given by the user with the variable *check_RWA_RF_stepsize*. If the condition is not fulfilled for any point in the volume, the respective new grid point is highlighted. An index file called *indexfile_check_RWA_RF.xslx* facilitates a comparison between the old and new indices. It is stored by default in the directory *Export_RWA_check*.

Listing 3.14 shows the creation process of the index file. The *columns* variable is a 1D array determining the number of initial grid points of one axis transformed into a new point. As an example, for the x-axis with 61 points and a step size of 10, the *colums* are given by $[10, 10, 10, 10, 10, 10, 1]$. The endpoints of each range are given by the cumulative sum of *columns* and are stored in the variable *x_end*. The starting point can be deduced from the endpoint using the formula $endpoint - stepsize + 1$. Only for the edge of the grid, the starting point is determined by the endpoint of the prior volume. The range of the new grid points in *mm* is given by the axis variables *x, y, z*. After generating a table with all the information, it is written to the aforementioned file.

The next step is the calculation of the RWA conditions. The first condition *detuning* $\ll$ *Larmor frequency* is shown as an example in listing 3.15. Both relevant parameters are calculated as 3D arrays, assigning a value to each grid point. The comparison with the $<$ operator yields a 3D logical array with only *1* and *0*. For the further procedure, its logical opposite is used, which makes it easier to plot the errors later. If the grid was coarsened before, the same must be done with the logical array. The built-in function *mat2cell* converts arrays to cell arrays. The *columns* variables determine the size of each cell array entry in terms of initial grid points. Applying the cell function *any* three times, once for each axis, yields *1* if any point in the cell array element is *1*, or else *0*. The built-in function *cell2mat* returns a now coarsened 3D logical array, which is plotted by the function *plot_logical_array_3D*. It returns the respective figure as frame.

The function *check_RWA_conditions* returns a 1D array containing the frame of each condition and their combined result.

```
function [B_eff_abs,frames] = effective_RF_field(B_static_abs,B_static_vec,
    B_RF,atom_state,frequency_RF,phases_RF,check_RWA_RF_properties,x,y,z,
    precision,current_step)
% Calculate the effective RF field by combining static and RF fields

% Parameters
% B_eff_abs                [out]: 3D array determining the absolute
%                                 effective magnetic field for every
%                                 grid point
% frames                   [out]: 1D array containing the frames from the
%                                 RWA check
% B_static_abs             [in] : 3D array determining the absolute static
%                                 magnetic field
% B_static_vec             [in] : 4D array determining the vectorial
%                                 static magnetic field
% B_RF                     [in] : 1D cell array containing a 4D array for
%                                 each wire. The 4D array is the magnetic
%                                 field amplitude vector of the respective
%                                 contribution to the RF field.
% atom_state               [in] : 1D array determining the atom state
%                                 [F,m_F]
% frequency_RF             [in] : Double/1D array determining the
%                                 frequency of the RF field
% phases_RF                [in] : 1D array determining the phases of the
%                                 RF field
% check_RWA_RF_properties [in] : 1D array containing the check_RWA_RF and
%                                 check_RWA_RF_stepsize variables
% x                        [in] : 1D array determining the grid points on
%                                 the x-axis
% y                        [in] : 1D array determining the grid points on
%                                 the y-axis
% z                        [in] : 1D array determining the grid points on
%                                 the z-axis
% precision                [in] : String determining the precision of the
%                               data
% current_step            [in] : Integer determining the current step of
%                               the simulation (1 for a single step
%                               simulation)
```

Listing 3.9: Function for calculating the effective RF magnetic field.

```
% Calculate rotation angles for the RF field with respect to the
% static magnetic field. Some of these angles need to be modified
% because otherwise there would be an unwanted phase shift of 180 degrees.
alpha = double(-atand(B_static_vec(:,:,:,2)./B_static_vec(:,:,:,1)));
alpha(B_static_vec(:,:,:,1)<0) = alpha(B_static_vec(:,:,:,1)<0) - 180;
alpha(isnan(alpha)) = 0;
beta = double(atand(-sqrt(B_static_vec(:,:,:,1).^2+B_static_vec(:,:,:,2).^2)
    ./B_static_vec(:,:,:,3)));
beta(B_static_vec(:,:,:,3)<0) = beta(B_static_vec(:,:,:,3)<0) - 180;
beta(isnan(beta)) = 0;
```

Listing 3.10: Calculate the rotation angles from the local direction of the static magnetic field.

```
% Get grid size
ranges = size(B_static_abs);

% Create empty cell for the rotated RF field
B_RF_rotated = cell(1, size(phases_RF,2));

for i_ChipWire = 1:size(B_RF,2)
    B_RF_rotated{i_ChipWire} = zeros(ranges(1),ranges(2),ranges(3),3,
        precision);
    % If the phase for this chip wire is NaN then the wire is not used for
    % the RF signal
    if ~isnan(phases_RF(1,i_ChipWire))

        % Rotation angle used for adding the contributions of the RF field
            with
        % respect to their relative phase
        delta = double(-(g_factor/abs(g_factor))*phases_RF(1,i_ChipWire));

        % Iterate over the whole grid to rotate every single RF vector in a
        % correct way
        for i_x = 1:ranges(1)
            for i_y = 1:ranges(2)
                for i_z = 1:ranges(3)
                    % Rotate the RF field so its z-component is parallel to
                    % the static field (which points in z-direction after
                        this
                    % transformation) and its x- and
                    % y-component are perpendicular to the static field.
                    % The last rotation around z is to add up the
                    % contributions of the RF field with respect to their
                    % relative phase.
                    B_RF_at_point = reshape(B_RF{i_ChipWire}(i_x,i_y,i_z,:),3,[]);
                    alpha_at_point = alpha(i_x,i_y,i_z);
                    beta_at_point = beta(i_x,i_y,i_z);
                    B_RF_rotated{i_ChipWire}(i_x,i_y,i_z,:) = rotz(delta)*roty(beta_at_point)*rotz(alpha_at_point)*B_RF_at_point;
                end
            end
        end
    end
end

% Sum over the different contributions to the RF field by the chip wires
B_RF_vec = sum(cat(5,B_RF_rotated{:}),5);
```

Listing 3.11: Calculate the rotated RF magnetic field.

```
% Calculate the effective field from the absolute static and the RF field
B_eff_abs = sqrt((squeeze(B_static_abs(:,:,:))-(h_bar*frequency_RF/abs(g_factor*bohr_magneton))*ones(ranges(1),ranges(2),ranges(3))).^2+0.25*(B_RF_vec(:,:,:,1).^2+B_RF_vec(:,:,:,2).^2));
```

Listing 3.12: Calculate the absolute effective RF magnetic field.

```
function frames = check_RWA_conditions(x,y,z,coarsening_stepsize,B_static_abs,B_oscillating_vec,oscillating_frequency,g_factor,m_F,indexfile_name,current_step)
% Check the RWA conditions for validity

% Parameters
% frames                [out]: 1D array containing the plots from the RWA
%                              check as frames
% x                     [in] : 1D array determining the grid points on
%                              the x-axis
% y                     [in] : 1D array determining the grid points on
%                              the y-axis
% z                     [in] : 1D array determining the grid points on
%                              the z-axis
% coarsening_stepsize   [in] : Integer determining the coarsening
%                              stepsize
% B_static_abs          [in] : 3D array determining the absolute static
%                              field
% B_oscillating_vec     [in] : 4D array determining the vectorial
%                              oscillating field
% oscillating_frequency [in] : Double determining the oscillating
%                              frequency
% g_factor              [in] : Double determining g_F of used atom state
% m_F                   [in] : Integer determining m_F of used atom state
% indexfile_name        [in] : String/Array determining filepath and
%                              -name of the index file
% current_step          [in] : Integer determining the current step of
%                              the simulation (1 for a single step
%                              simulation)
```

Listing 3.13: Function for checking the RWA conditions.

```
% Create columns
columns_x = [ones(1,fix(ranges(1)/coarsening_stepsize))*coarsening_stepsize
    rem(ranges(1),coarsening_stepsize)];

length_of_longest_array = max([length(nonzeros(columns_x))]);
% new indices of coarse grid
index = (1:length_of_longest_array)';

% Initialize all variables for the table: Every element has to have the
% same amount of rows therefore they are initialized with NaNs using
% length_of_longest_array
x_start_end = NaN(length_of_longest_array,2);
x_range_in_mm = NaN(length_of_longest_array,2);

% Calculate the start and end of the original indices for each new
% index (index range)
x_end = cumsum(nonzeros(columns_x));
x_start = x_end - coarsening_stepsize + 1;

% If the coarsening_stepsize is smaller than the amount of indices for the
% respective axis then adapt the last entry. If the coarsening_stepsize
    equals the
% amount of indices, this is not necessary since there will be only one
% index here. The coarsening_stepsize can't be bigger than the smallest
    amount of
% indices in any direction.
if coarsening_stepsize<ranges(1)
    x_start(end) = x_end(end-1)+1;
end

% Write the index and axis ranges to their correctly sized variables
x_start_end(1:length(nonzeros(columns_x)),:) = [x_start x_end];
x_range_in_mm(1:length(nonzeros(columns_x)),:) = x([x_start x_end])*1e3;

% Write the variables to a table and save it as indexfile_name
t = table(index, x_start_end(:,1),x_start_end(:,2),x_range_in_mm(:,1),
    x_range_in_mm(:,2));
t.Properties.VariableNames = {'New_Index', 'x_Index_Start','x_Index_End', '
    x_Range_Start_in_mm','x_Range_End_in_mm'};
writetable(t,indexfile_name,'WriteMode','overwritesheet');
```

Listing 3.14: Example of generating an index file for RWA checks (only for one axis).

```
% Calculate new coordinates for coarse grid
if coarsening_stepsize > 1
    [x_rwa,y_rwa,z_rwa] = ndgrid(1:length(columns_x),1:length(columns_y),1:
        length(columns_z));
```

```
else
    [x_rwa,y_rwa,z_rwa] = ndgrid(1:ranges(1),1:ranges(2),1:ranges(3));
end

% Larmor frequency
freq_larmor = abs(bohr_magneton*g_factor)*squeeze(B_static_abs(:,:,:))/h_bar
    ;

% First RWA condition: Detuning << Larmor frequency
detuning = abs(abs(bohr_magneton*g_factor)*squeeze(B_static_abs(:,:,:))/
    h_bar-oscillating_frequency*ones(ranges(1),ranges(2),ranges(3)));
detuning_check = freq_larmor > detuning;
if coarsening_stepsize>1
    cell_check = mat2cell(~detuning_check, columns_x,columns_y,columns_z);
    cell_check_or = cellfun(@(x) any(any(any(x))), cell_check,'UniformOutput
        ',0);
    detuning_check_coarse = cell2mat(cell_check_or);
else
    detuning_check_coarse = ~detuning_check;
end

% Plot the result
frames(1) = plot_logical_array_3D(x_rwa,y_rwa,z_rwa,detuning_check_coarse,'
    RWA check: Detuning << Larmor frequency',coarsening_stepsize);
```

Listing 3.15: Example of checking one RWA condition.

Calculation of the potential energy

Now the potential energy can be calculated by multiplying $g_F \mu_B \tilde{m}_F$ to the effective magnetic field (equation 2.31). This works for both static and RF fields and is done by the function *calculate_potential_energy*. Details are shown in listing 3.16. The function contains an additional algorithm for adding gravity to the potential if the user has enabled this setting in listing 3.1.

```
function [atom_pot,i_atom_pot_min,effective_pot_min] =
    calculate_potential_energy(addpot,atom_state,pot_unit,B_eff_abs,U_dipole
    ,z)
% Calculate the potential energy from the magnetic field

% Parameters
% atom_pot          [out]: 3D array determining the potential energy of
%                          each grid point
% i_atom_pot_min    [out]: 1D array containing indices of potential
%                          minimum
% effective_pot_min [out]: Double determining potential minimum
% addpot            [in] : Struct determining the different contributions
%                          to the magnetic field
% atom_state        [in] : 1D array determining the atom state [F,m_F]
% pot_unit          [in] : String determining the potential energy unit
%                          (J or K)
% B_eff_abs         [in] : 3D array determining the absolute effective
%                          field
% U_dipole          [in] : Cell array determining the dipole potential
% z                 [in] : 1D array determining the grid points on the
%                          z—axis
```

Listing 3.16: Function for calculating the potential energy from the effective field.

Difference between single-step and sequence simulation

The only real difference between a single-step and a sequence simulation can be found in the function *calculate_magnetic_field*. For a single-step simulation, the functions mentioned above are called one after the other. The RWA checks are plotted instantly. On the other hand, a sequence is simulated in a parallel *for*, a *parfor* loop, which has some special requirements in comparison to a normal *for* loop. Variables that are accessed by every iteration step have to be initialized with the correct size before the loop. Furthermore, plotting during the execution doesn't work, so the figures are returned as frames by the respective functions and are stored in the directory *Export_RWA_check* as videos with one frame for each simulation step.

3.4 Control of the waveform generator

In order to apply the theory in practice, two different RF signals are needed, which are best provided by a dual-channel waveform generator. In this experiment, a Tabor WW5062 is used. Further details on the device are listed in section 3.1.4.

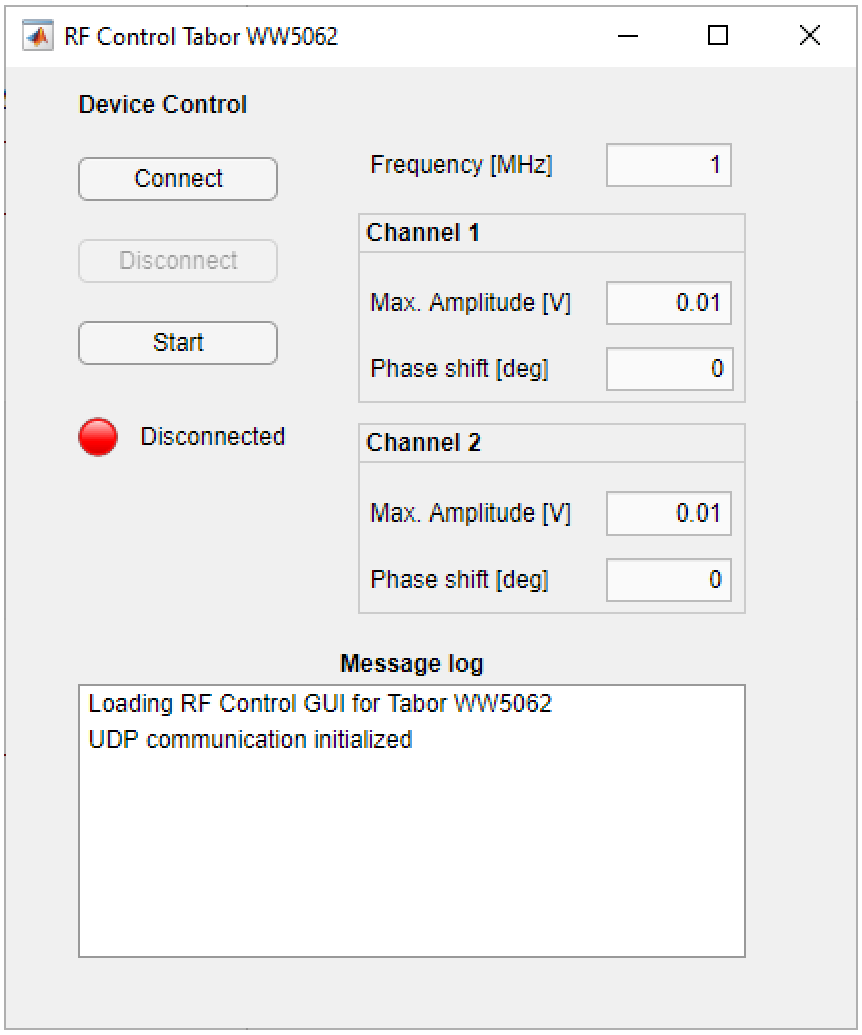
This chapter provides the control software for the waveform generator, which functions as the connection between the user input and the device itself. The former is provided via a measurement and data processing system called *ADwin*, for which the code has already existed before this thesis. The software presented here communicates with the *ADwin* user interface and uses the built-in MATLAB functions for accessing the parameters of the Tabor device. The graphical user interface (GUI) of the Tabor control software is shown in figure 3.7 and does not allow for manually setting any parameters there.

The internal properties of the software are shown in listing 3.17.

When starting the control software, the function *Open_GUI* is called. It initializes a UDP connection to the *ADwin* and loads initial values for the parameters from an external file, which are shown in the window of figure 3.7. The function is shown in listing 3.18.

The button *Connect* calls the function *Push_Connect* (see listing 3.19), which connects the software with the Tabor WW5062 via Ethernet. The initial values are now provided to the waveform generator. The amplitude and phase have to be set separately for each channel, the frequency needs to be specified only once, no matter which channel is currently active.

The measurement is started by clicking the button *Start*, which activates the function *Toggle_Start* shown in listing 3.20. It waits for commands from the *ADwin* and applies the received settings to the waveform generator. The *ADwin* sends a string containing the frequency, amplitudes and phases for both channels, each



```
Loading RF Control GUI for Tabor WW5062
UDP communication initialized
```

Figure 3.7: Window of Tabor WW5062 control software.

separated by a tilde. After adjusting the values, the function waits for a new string. If the string contains *Close_GUI*, the function is stopped and the GUI is closed. If *Pause* is the first element of that string, the amplitudes are set to their initial value. In this case, the outputs stay on and the signal is directed to a 50 Ohm resistance via an RF switch.

After the measurements, the waveform generator can be disconnected from the software by pushing the button *Push_Disconnect*. All the parameters are then set back to the initial values

```
classdef RF_WW5062 < matlab.apps.AppBase
    properties (Access = private)

            % Identical to root path in Adwin default parameters
            root_path = 'C:\Users\Caesium\ADwin\';

            % RF settings
            Freq_init
            Amp_1_init
            Phase_1_init
            Amp_2_init
            Phase_2_init

            Frequency
            Amplitude_1
            Phase_1
            Amplitude_2
            Phase_2

            % Device communication
            rfSource = 1; % initialize as double to allow for checking
                connection state before first connection
            groupConf = 1;

            % UDP communication
            udpReceiver

    end
end
```

Listing 3.17: Internal properties of the Tabor WW5062 control software.

```
function Open_GUI(app)

            instrreset

            str = 'Loading RF Control GUI for Tabor WW5062';
            disp(str)
            app.list_MessageLog.Items{1} = str;
            drawnow
            scroll(app.list_MessageLog,'bottom');

            % Create udp object
            IPControl = '128.131.60.61';
            PortSender = 9099;
            PortReceiver = 9100;
            app.udpReceiver = udp(IPControl,PortSender,'LocalPort',
                PortReceiver,'Timeout',0.1);

            str = 'UDP communication initialized';
            disp(str)
            app.list_MessageLog.Items{end+1} = str;
            drawnow
            scroll(app.list_MessageLog,'bottom');

            % Load file with initial sweep settings
            load([app.root_path,'Devices\RF_WW5062\ww5062_init.mat']);

            % Get initial values
            app.Freq_init = ww5062_init.freq;
            app.Amp_1_init = ww5062_init.amp_1;
            app.Phase_1_init = ww5062_init.phase_1;
            app.Amp_2_init = ww5062_init.amp_2;
            app.Phase_2_init = ww5062_init.phase_2;

            % Set to initial values
            Reset_parameters(app)

            % Update numeric indicators
            Update_numbers(app)

        end
```

Listing 3.18: Function called upon start-up of the control software.

```
function Push_Connect(app, event)

            str = 'Connecting...';
            disp(str)
            app.list_MessageLog.Items{end+1} = str;
            drawnow
            scroll(app.list_MessageLog,'bottom');

            % Connect to device
            app.rfSource = icdevice('ww107x_64.mdd','TCPIP::10.0.2.7::23::
                SOCKET');
            connect(app.rfSource);

            % Reset to initial values
            Reset_parameters(app)

            % Turn on output of both channels
            app.groupConf = get(app.rfSource, 'Configuration');
            invoke(app.groupConf, 'configureoutputenabled', 'CHAN_A', 1);
            invoke(app.groupConf, 'configureoutputenabled', 'CHAN_B', 1);

            % Set initial values to Tabor WW5062
            set(app.rfSource, 'RepCapIdentifier','CHAN_A'); % set Channel 1
                as active
            set(app.rfSource.Standardfunctionoutput(1), 'Waveform', 1); %
                select sine
            set(app.rfSource.Standardfunctionoutput(1),'Frequency', app.
                Frequency); % uniform and therefore only set once
            set(app.rfSource.Standardfunctionoutput(1),'Amplitude', app.
                Amplitude_1);
            set(app.rfSource.Standardfunctionoutput(1),'Start_Phase', app.
                Phase_1);

            set(app.rfSource, 'RepCapIdentifier','CHAN_B'); % set Channel 2
                as active
            set(app.rfSource.Standardfunctionoutput(1), 'Waveform', 1); %
                select sine
            set(app.rfSource.Standardfunctionoutput(1),'Amplitude', app.
                Amplitude_2);
            set(app.rfSource.Standardfunctionoutput(1),'Start_Phase', app.
                 Phase_2);


            app.button_Connect.Enable = 0;
            app.button_Disconnect.Enable = 1;
            app.button_Start.Enable = 1;
            app.boolean_OperationState.Color = 'b';
            app.label_OperationState.Text = 'Ready';

            str = 'Connection to RF source established';
            disp(str)
            app.list_MessageLog.Items{end+1} = str;
            drawnow
            scroll(app.list_MessageLog,'bottom');

        end
```

Listing 3.19: Callback function of the button *Connect*.

```
function Toggle_Start(app, event)

            drawnow
            value = app.button_Start.Value;

            if ~value

                fclose(app.udpReceiver);

                str = 'UDP communication closed';
                disp(str)
                app.list_MessageLog.Items{end+1} = str;
                drawnow
                scroll(app.list_MessageLog,'bottom');

                app.button_Disconnect.Enable = 1;
                app.button_Start.Text = 'Start';
                app.boolean_OperationState.Color = 'b';
                app.label_OperationState.Text = 'Ready';
            else
                % Initialize UDP session
                flushinput(app.udpReceiver); % empty UDP buffer
                fopen(app.udpReceiver);
                warning('off','instrument:fscanf:unsuccessfulRead') % this
                    turns off the annoying timeout warnings
                message = '';

                str = 'Ready for ADwin instructions';
                disp(str)
                app.list_MessageLog.Items{end+1} = str;
                drawnow
                scroll(app.list_MessageLog,'bottom');

                app.button_Connect.Enable = 0;
                app.button_Disconnect.Enable = 0;
                app.button_Start.Text = 'Stop';
                app.boolean_OperationState.Color = 'g';
                app.label_OperationState.Text = 'Waiting';

                while value

                    drawnow
                    value = app.button_Start.Value;
                    if ~value
                        break
                    end

                    % Wait for UDP message
                    while isempty(message)

                        drawnow
                        value = app.button_Start.Value;
                        if ~value
                            return
                        end

                        % Read UDP buffer
                        message = fscanf(app.udpReceiver);

                    end

                    str = ['Setting update: ',message];
                    disp(str)
                    app.list_MessageLog.Items{end+1} = str;
                    drawnow
                    scroll(app.list_MessageLog,'bottom');

                    % Convert message to values
                    input = strsplit(message,'~');
                    message = '';

                    % Break while loop via ADwin command
                    if strcmp(input{1},'Close_GUI')
                        break
                    % Reset amplitude if cycle is paused
                    elseif strcmp(input{1},'Pause')
                        app.Amplitude_1 = app.Amp_1_init;
                        app.Amplitude_2 = app.Amp_2_init;
                    % Set settings according to message
                    else
                        app.Frequency = str2double(input{1});
                        app.Amplitude_1 = str2double(input{2});
                        app.Phase_1 = str2double(input{3});
                        app.Amplitude_2 = str2double(input{4});
                        app.Phase_2 = str2double(input{5});
                    end

                    % Update numeric indicators
                    Update_numbers(app)

                    app.button_Start.Enable = 0;
                    app.boolean_OperationState.Color = 'r';
                    app.label_OperationState.Text = 'Transferring...';
                    drawnow

                    % Set values to Tabor WW5062
                    set(app.rfSource, 'RepCapIdentifier','CHAN_A'); % set Channel 1 as active
                    set(app.rfSource.Standardfunctionoutput(1), 'Waveform', 1); % select sine
                    set(app.rfSource.Standardfunctionoutput(1),'Frequency', app.Frequency);
                    set(app.rfSource.Standardfunctionoutput(1),'Amplitude', app.Amplitude_1);
                    set(app.rfSource.Standardfunctionoutput(1),'Start_Phase', app.Phase_1);

                    set(app.rfSource, 'RepCapIdentifier','CHAN_B'); % set Channel 2 as active
                    set(app.rfSource.Standardfunctionoutput(1), 'Waveform', 1); % select sine
                    set(app.rfSource.Standardfunctionoutput(1),'Amplitude', app.Amplitude_2);
                    set(app.rfSource.Standardfunctionoutput(1),'Start_Phase', app.Phase_2);

                    app.button_Start.Enable = 1;
                    app.boolean_OperationState.Color = 'g';
                    app.label_OperationState.Text = 'Waiting';
                    drawnow

                end

                % Close GUI via ADwin command
                if strcmp(input{1},'Close_GUI')
                    Close_GUI(app)
                end

            end
        end
```

Listing 3.20: Callback function of the button *Start*.

Amplitude Modulation (AM) Input

By slowly changing the amplitude of the RF field, the emergence of dressed potentials can be achieved. The Tabor WW5062 does not offer a digital amplitude ramp function, but provides an analog AM input. By applying a voltage between 0 V and 5 V, the set amplitude is multiplied by a factor between 1 and 0. With zero applied voltage, the factor is 1, for 5 V peak-to-peak voltage, the factor is almost 0. Unfortunately, there is no function to enable the output modulation mode and amplitude modulation in particular with MATLAB. A possible workaround is changing the settings directly on the device. They can be found under $TOPMenu \rightarrow Waveform \rightarrow Modulated \rightarrow Modulation\,Type \rightarrow AM$. This has to be done separately for both channels.

4 Results

Here, the results of the potential calculation for a Ioffe-Pritchard-type trap are presented. A quadrupole-like trap is not recommended because of the vanishing field in the trap center. The general and static field parameters are displayed in listings 3.1 and 3.2, the RF configuration is given for each result separately. The details regarding the relation of detuning and potential shape are mainly summarized from [8].

An example of possible potential shapes achievable with RF dressing is shown in figure 4.1.

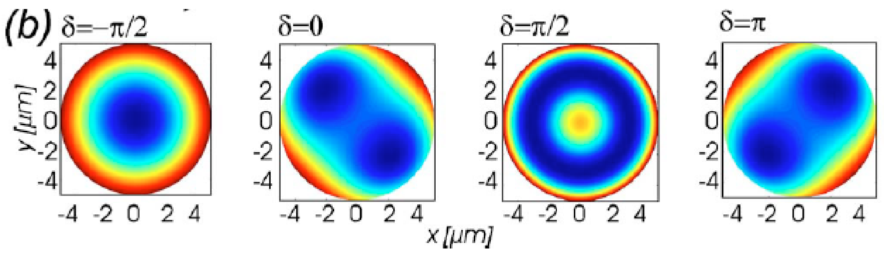


Figure 4.1: Examples for possible potential shapes: Single-well, double-well, ring [11].

Every configuration will be shown by the following figures:

- 3D plot of an equipotential surface
- 2D plots along axes through potential minimum
- 1D plots along axes through potential minimum
- First RWA condition: Detuning < Larmor frequency
- Second RWA condition: Rabi frequency < Larmor frequency

4.1 Linear polarization: Double-well

Phase shifts of $\gamma = 0, \pi$ describe a linearly polarized field. Depending on the detuning Δ and the total absolute RF field amplitude, two potential shapes are possible. If $\Delta < 0$ and the absolute RF field amplitude is below a critical field strength, which depends on the static field and the detuning, the static potential is simply deformed. In any other case, the result is a double-well potential, which is rotated by 90 degrees for a phase shift $\gamma = \pi$ in comparison to $\gamma = 0$.

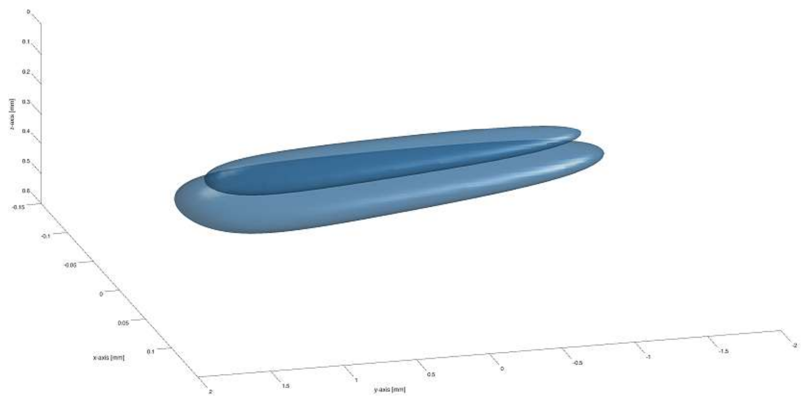
The simulation results for a phase shift $\gamma = 0$ are shown in figure 4.2, the used parameters are listed in listing 4.1. The ones for $\gamma = \pi$ with the RF configuration in listing 4.2 can be seen in figure 4.3

```
% RF current amplitudes in A (steps possible)
OuterTrap1_amp_RF = -1;
OuterTrap2_amp_RF = 1;

% RF frequency in Hz (normal frequency) (steps possible)
frequency_RF = 5e5;

% RF phases in rad (steps not allowed)
OuterTrap1_phase_RF = 0;
OuterTrap2_phase_RF = 0;

check_RWA_RF = 1;
check_RWA_RF_stepsize = 5;
```

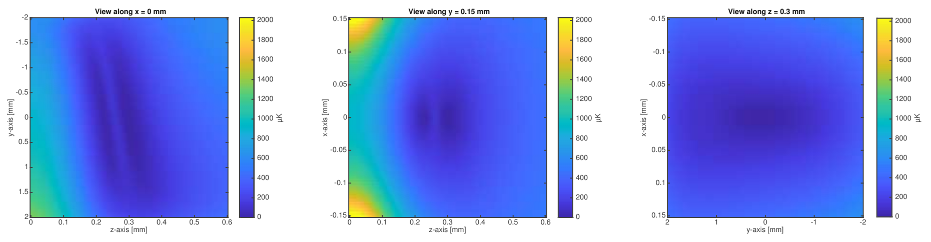
Listing 4.1: RF configuration for double-well potential shown in figure 4.2.

```
% RF current amplitudes in A (steps possible)
OuterTrap1_amp_RF = -0.5;
OuterTrap2_amp_RF = 0.5;

% RF frequency in Hz (normal frequency) (steps possible)
frequency_RF = 7.5e5;

% RF phases in rad (steps not allowed)
OuterTrap1_phase_RF = 0;
OuterTrap2_phase_RF = pi;

check_RWA_RF = 1;
check_RWA_RF_stepsize = 5;
```

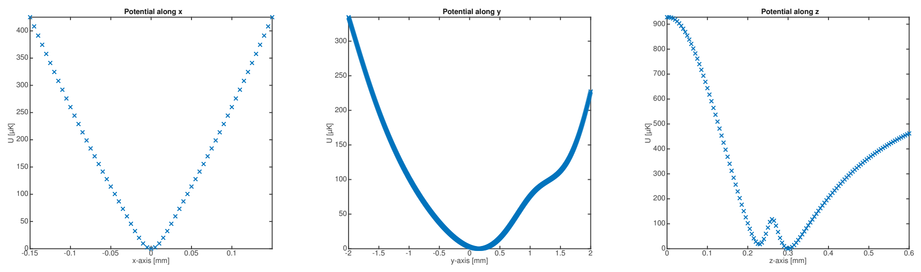
Listing 4.2: RF configuration for double-well potential shown in figure 4.3.



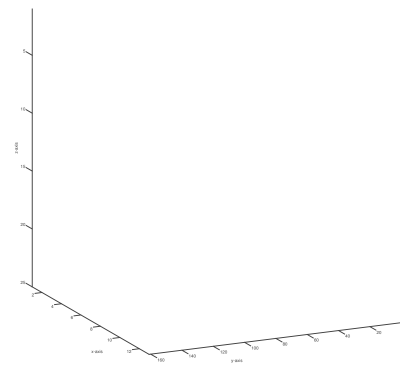
(a) Equipotential surface of 100 µK around minimum.



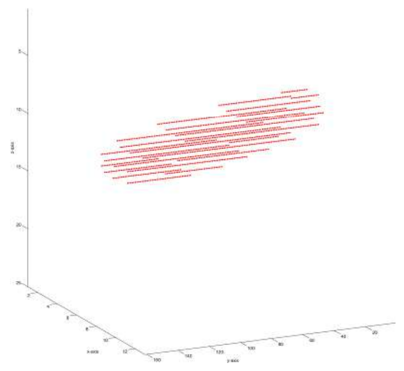
(b) 2D plots along axes through minimum.



(c) 1D plots along axes through minimum.

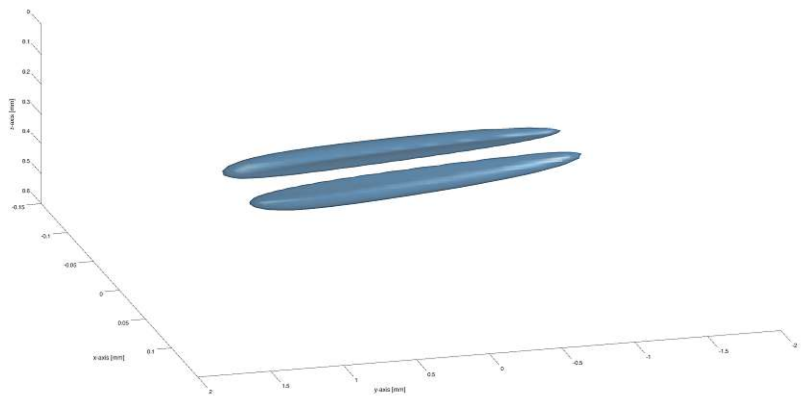


(d) RWA condition:
Detuning < Larmor frequency.

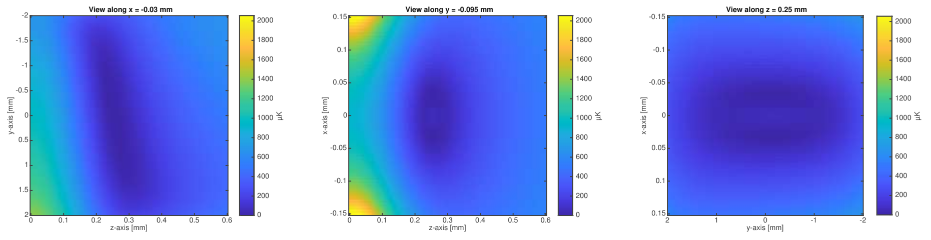


(e) RWA condition:
Rabi frequency < Larmor frequency.

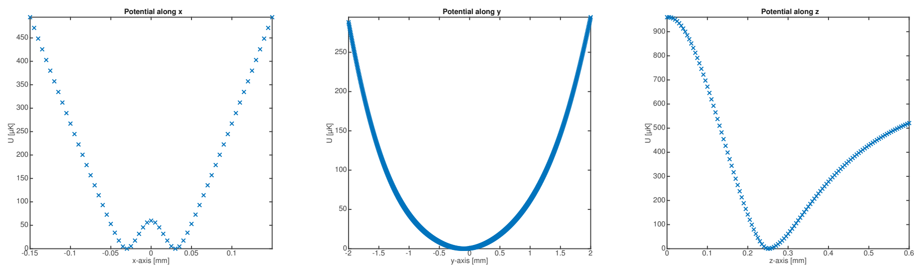
Figure 4.2: Double-well potential for a phase shift of $\gamma = 0$: minimum is at 126 µK. Parameters are given by listing 4.1.



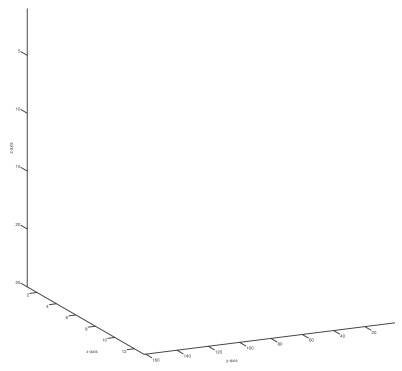
(a) Equipotential surface of 25 µK around minimum.



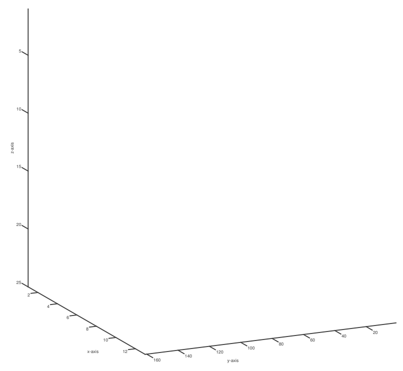
(b) 2D plots along axes through minimum.



(c) 1D plots along axes through minimum.



(d) RWA condition:
Detuning $<$ Larmor frequency.



(e) RWA condition:
Rabi frequency $<$ Larmor frequency.

Figure 4.3: Double-well potential for a phase shift of $\gamma = \pi$: minimum is at 38.8 µK. Parameters are given by listing 4.2.

4.2 Circular polarization: Single-well and ring

A circularly polarized RF field can be achieved with phase shifts $\gamma = \pi/2, 3\pi/2$ and with identical absolute currents in both wires. If the RF frequency is bigger than the Larmor frequency and thus $\Delta > 0$, the potential is always ring-shaped. If $\Delta < 0$, the result is either a single-well or a ring, depending on the sign of the Landé factor g_F. For the parameters used, a phase shift of $\gamma = \pi/2$ yields a ring trap, while $\gamma = 3\pi/2$ gives a single-well potential. The first configuration is shown in figure 4.5, its RF configuration is given by listing 4.4. In order to characterize the dressed single-well trap, the trap frequencies (cf. equation 2.12) were determined by applying a harmonic fit function to each grid point with a potential $\leq trap bottom + 5\,\mu K$. The results are shown in table 4.1. The results for a ring potential are displayed in figure 4.4, its parameters can be found in listing 4.3.

```
% RF current amplitudes in A (steps possible)
OuterTrap1_amp_RF = -0.4;
OuterTrap2_amp_RF = 0.4;

% RF frequency in Hz (normal frequency) (steps possible)
frequency_RF = 9e5;

% RF phases in rad (steps not allowed)
OuterTrap1_phase_RF = 0;
OuterTrap2_phase_RF = pi/2;

check_RWA_RF = 1;
check_RWA_RF_stepsize = 5;
```

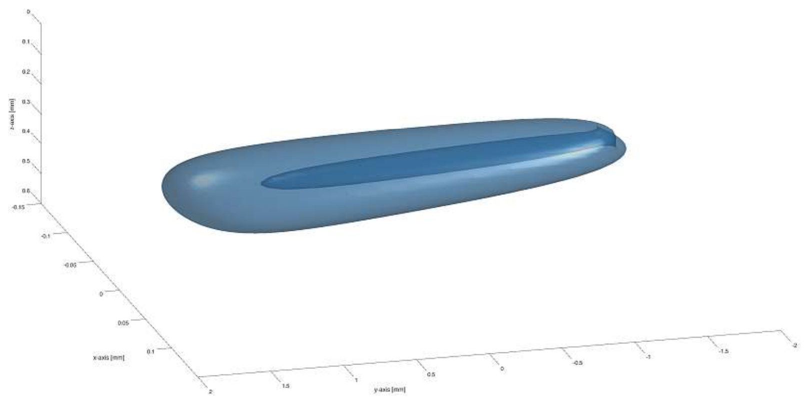
Listing 4.3: RF configuration for ring potential shown in figure 4.4.

ν_x [Hz]	ν_y [Hz]	ν_z [Hz]
637	12.5	287

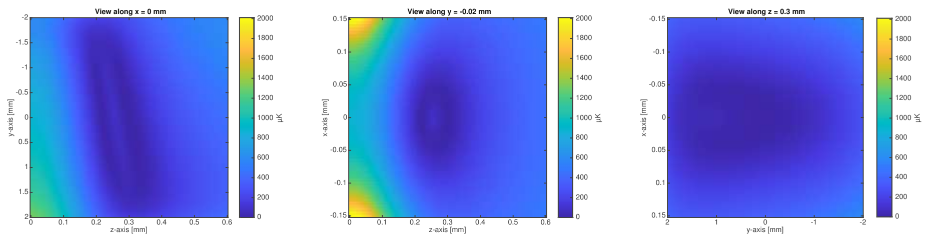
Table 4.1: Trap frequencies (cf. equation 2.12) of the single-well potential shown in figure 4.5c.

```
% RF current amplitudes in A (steps possible)
OuterTrap1_amp_RF = -0.4;
OuterTrap2_amp_RF = 0.4;

% RF frequency in Hz (normal frequency) (steps possible)
frequency_RF = 3e5;

% RF phases in rad (steps not allowed)
OuterTrap1_phase_RF = 0;
OuterTrap2_phase_RF = 3*pi/2;

check_RWA_RF = 1;
check_RWA_RF_stepsize = 5;
```

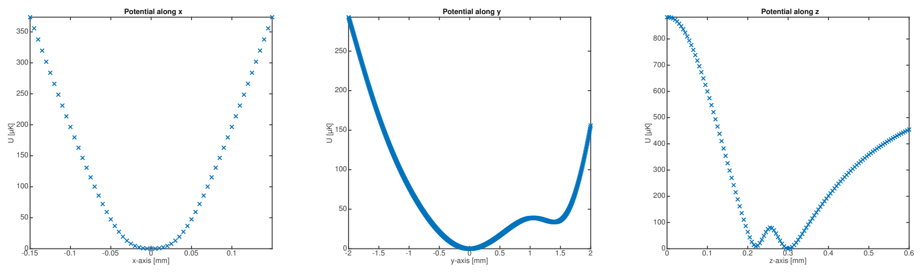
Listing 4.4: RF configuration for single well potential shown in figure 4.5.



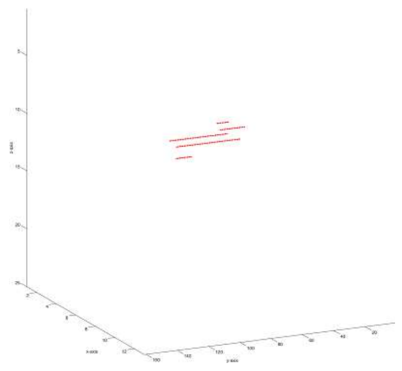
(a) Equipotential surface of 50 µK around minimum.

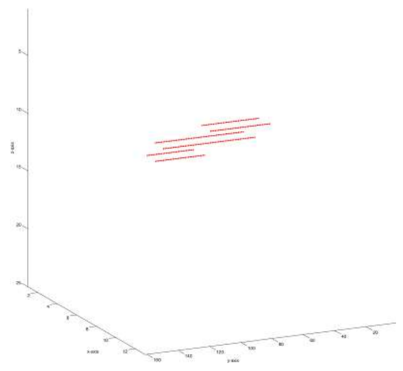


(b) 2D plots along axes through minimum.



(c) 1D plots along axes through minimum.

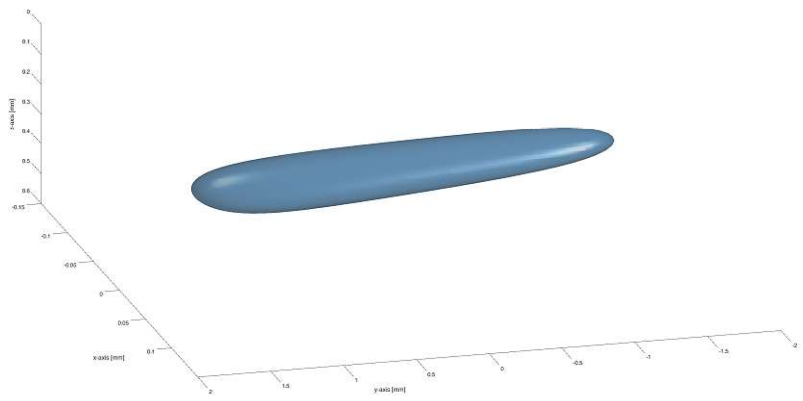




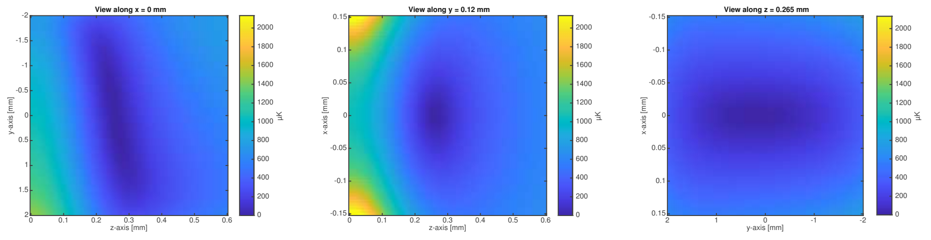
(d) RWA condition:
Detuning < Larmor frequency.

(e) RWA condition:
Rabi frequency < Larmor frequency.

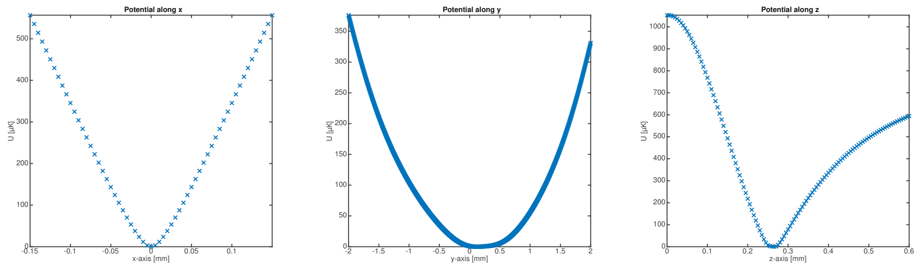
Figure 4.4: Ring potential for a phase shift of $\gamma = \pi/2$: minimum is at 69.5 µK. Parameters are given by listing 4.3.



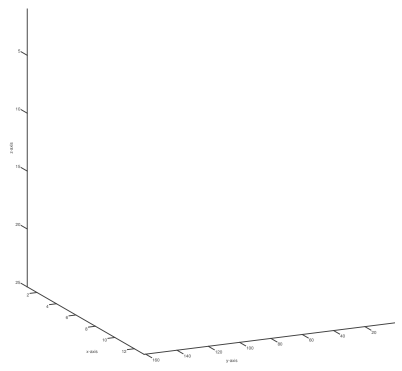
(a) Equipotential surface of 100 µK around minimum.



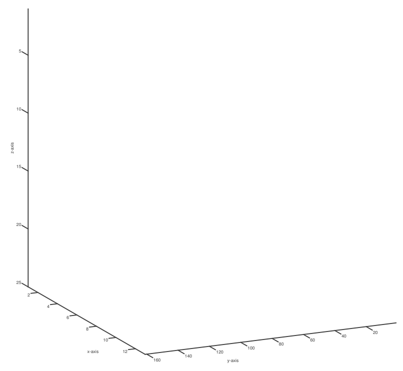
(b) 2D plots along axes through minimum.



(c) 1D plots along axes through minimum.



(d) RWA condition:
Detuning < Larmor frequency.



(e) RWA condition:
Rabi frequency < Larmor frequency.

Figure 4.5: Single-well potential for a phase shift of $\gamma = 3\pi/2$: minimum is at 30.2 µK. Parameters are given by listing 4.4.

4.3 Radio-frequency evaporative cooling

Trapping atoms requires sufficiently cold atoms. Otherwise, the atoms have too much energy and can be lost from the trap. A possible approach is removing hot atoms selectively from the trap and thus keeping the cold ones only. This is called evaporative cooling and can be done using a radio-frequency magnetic field. An external static magnetic field lifts the degeneracy of the atom states in m_F. In a magnetic trap, the cold atoms are located around the center while the hot atoms can move further outwards due to their kinetic energy. Figure 4.6a shows an example of three different m_F states that are coupled by an RF field. The coupling effectively reintroduces the degeneracy in m_F at the respective spatial points, transferring atoms to magnetically non-trappable states. Since the RF frequency determines the distance of the coupling points A and A' from the trap center (or the radius of a coupling ring in three dimensions), hot atoms can be removed from the trap with a sufficiently high frequency. After a ramp from high to low frequencies, only the cold atoms remain in the trap. [18]

The simulation can be used to determine the optimal wire configurations for RF cooling. The plots in figure 4.7 show the potential energy in 2D along the axes through the minimum for different RF frequencies. The used configuration is the chip quadrupole trap (cf. section 3.1.3); the outer trap wires (cf. table 3.4) are used for the RF signal with an amplitude of 16 mA and a phase shift of zero degrees.

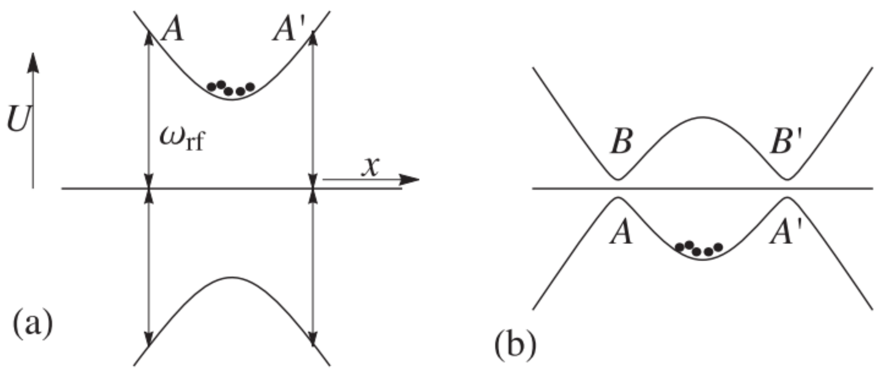
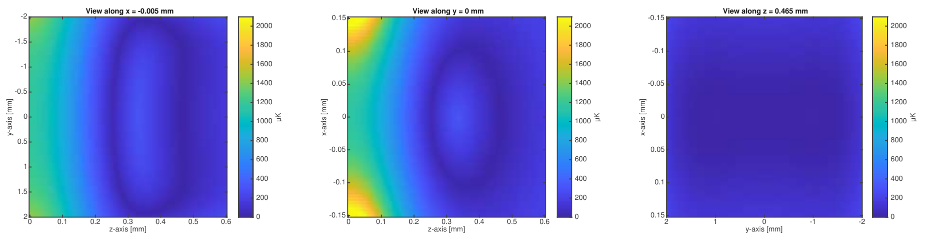
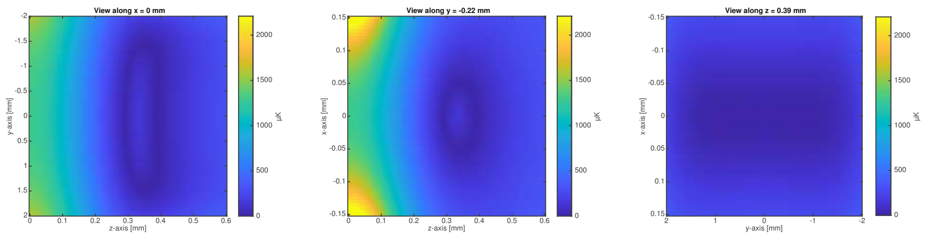


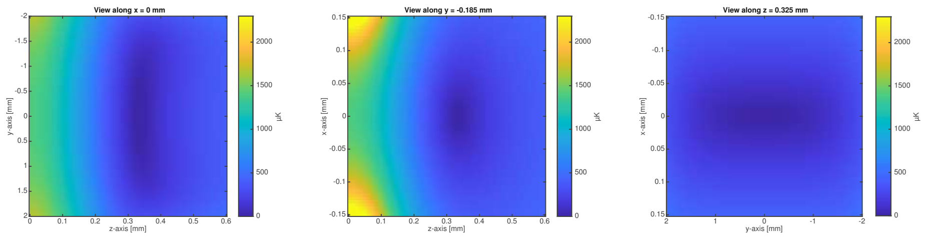
Figure 4.6: RF cooling in a magnetic trap: coupling of different m_F states (a) leads to an effective potential (b) [19].



(a)



(b)



(c)

Figure 4.7: 2D plots of dressed potentials for frequencies of (a) 10 MHz, (b) 5 MHz and (c) 1 MHz.

Bibliography

[1] Bose. Plancks Gesetz und Lichtquantenhypothese. *Zeitschrift fur Physik*, 26(1):178–181, December 1924.

[2] Albert Einstein. *Quantentheorie des einatomigen idealen Gases*. de Gruyter, Berlin, 1925. [Sonderabdruck aus: Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-Mathematische Klasse. 1925, S. 3-14].

[3] M. H. Anderson, J. R. Ensher, M. R. Matthews, C. E. Wieman, and E. A. Cornell. Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor. 269:198–201, 1995.

[4] Tino Weber. *Bose-Einstein Condensation of Optically Trapped Cesium*. Phd thesis, Leopold-Franzens-Universität Innsbruck, September 2003.

[5] O. Zobay and B. M. Garraway. Two-dimensional atom trapping in field-induced adiabatic potentials. *Physical Review Letters*, 86(7):1195–1198, February 2001.

[6] Daniel A. Steck. Cesium D Line data, 1998.

[7] T. Kraemer, J. Herbig, M. Mark, T. Weber, C. Chin, H.-C. Nägerl, and R. Grimm. Optimized production of a cesium Bose–Einstein condensate. 79:1013–1019, 2004.

[8] Sebastian Hofferberth. *Coherent manipulation of Bose-Einstein condensates with radio-frequency adiabatic potentials on atom chips*. Phd thesis, Ruperto-Carola University of Heidelberg, July 2007.

[9] Ettore Majorana. Atomi orientati in campo magnetico variabile. 9:43–50, 1932.

[10] David E. Pritchard. Cooling neutral atoms in a magnetic trap for precision spectroscopy. *Physical Review Letters*, 51(15):1336–1339, October 1983.

[11] I. Lesanovsky, T. Schumm, S. Hofferberth, L. M. Andersson, P. Krüger, and J. Schmiedmayer. Adiabatic radio-frequency potentials for the coherent manipulation of matter waves. *Physical Review A: General Physics*, 73(3):033619, March 2006.

[12] Evan Ali Salim. *Ultracold matter systems and atomtronics instrumentation*. phdthesis, University of Colorado, 2011.

[13] K. Brugger, P. Krüger, X. Luo, S. Wildermuth, H. Gimpel, M. W. Klein, S. Groth, R. Folman, I. Bar-Joseph, and J. Schmiedmayer. Two-wire guides and traps with vertical bias fields on atom chips. *Physical Review A: General Physics*, 72:023607, August 2005.

[14] ColdQuanta Inc. Atom chip delivery sheet, 2018.

[15] Tabor Electronics Inc. Data sheet for model WW5062, April 2021.

[16] Clemens Maderböck. Simulation von Magnetfeldern einer Quadrupolfalle für Cäsiumatome, October 2017.

[17] S. Hofferberth, B. Fischer, T. Schumm, J. Schmiedmayer, and I. Lesanovsky. Ultracold atoms in radio-frequency dressed potentials beyond the rotating-wave approximation. *Physical Review A: General Physics*, 76(1):013401, July 2007.

[18] O. Morizot, C. L. Garrido Alzar, P.-E. Pottie, V. Lorent, and H. Perrin. Trapping and cooling of rf-dressed atoms in a quadrupole magnetic field. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 40:4013–4022, 2007.

[19] Barry M. Garraway and Hélène Perrin. Recent developments in trapping and manipulation of atoms with adiabatic potentials. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 49:172001, 2016.