Degenerate Fermi Gas of ^{87}Sr

B.J. DeSalvo

Rice University, Department of Physics and Astronomy, Houston, Texas, 77251

(Dated: April 10, 2012)

Degenerate Fermi Gas of ^{87}Sr

B. J. DeSalvo, M. Yan, P. G. Mickelson, Y. N. Martinez de Escobar, and T. C. Killian
Department of Physics and Astronomy, Rice University, Houston, Texas, 77251, USA
(Received 5 May 2010; published 13 July 2010)

We report quantum degeneracy in a gas of ultracold fermionic ^{87}Sr atoms. By evaporatively cooling a mixture of spin states in an optical dipole trap for 10.5 s, we obtain samples well into the degenerate regime with $T/T_F = 0.26^{+0.05}_{-0.06}$. The main signature of degeneracy is a change in the momentum distribution as measured by time-of-flight imaging, and we also observe a decrease in evaporation efficiency below $T/T_F \sim 0.5$.

DOI: 10.1103/PhysRevLett.105.030402 PACS numbers: 03.75.Ss

Quantum degeneracy of fermions in dilute atomic gases [1] is currently one of the most active areas of physics research. Tunable interactions [2] have allowed an exploration of the Bose-Einstein condensation (BEC)-BCS crossover [3] as well as the study of phenomena in the unitary regime such as superfluidity in systems with [4,5] and without [6–8] spin imbalance. Research on Fermi gases in optical lattices [9] can make direct connections to the properties of electrons in solids and realize important models like the Hubbard Hamiltonian [10], but with new capabilities for controlling system parameters such as density, interaction strength, and dimensionality. Current searches are underway for analogues of high temperature superconductivity [11] and spontaneous Néel magnetism [12].

Here we report quantum degeneracy of the fermionic isotope of strontium ^{87}Sr. Five other species of fermions have been brought into the quantum degenerate regime (^{40}K [13], ^{6}Li [14], ^{3}He [15], ^{171}Yb, and ^{173}Yb [16]), but ^{87}Sr has several properties which make quantum degenerate samples of this type particularly interesting. ^{87}Sr has a very large nuclear spin, $I = 9/2$, which may allow studies of novel magnetic phenomena due to enlarged SU(N) symmetry of the interaction Hamiltonian for $N = 2I + 1$ [17–20]. High resolution spectroscopy technologies are the most advanced in strontium because of the use of narrow intercombination transitions in ^{87}Sr for optical frequency standards [21], and these tools have motivated proposals for applications in quantum information [22–24] and quantum simulation of many-body phenomena [18,25]. Strontium also has a number of stable bosons which have recently been brought into the quantum degenerate regime [26–28], which makes Fermi-Bose mixtures with relatively small mass differences available [29]. There is also the potential for manipulating interactions on small spatial and temporal scales with low-loss optical Feshbach resonances [30,31].

Details about our apparatus can be found in [27,32,33]. Atoms are trapped from a Zeeman slowed beam in a magneto-optical trap (MOT) operating on the $(5s^2)^1S_0 - (5s5p)^1P_1$ transition at 461 nm (Fig. 1). Since this transition is not closed, approximately 1 in 10^5 excitations results in an atom decaying through the $(5s5d)^1D_2$ state to the $(5s5p)^3P_2$ state, which has a 9 min lifetime [34] and can be trapped in the quadrupole magnetic field of the MOT [35–38]. The magnetic trap has a lifetime of about 25 s, limited by background pressure and blackbody radiation [37]. This allows us to accumulate atoms over a 30 s period and trap a significant number in spite of the low natural abundance of ^{87}Sr (7%).

After accumulation, atoms in the $^3P_2$ state are returned to the ground state via excitation to the $(5s4d)^3D_2$ state [33]. This repumping is achieved by applying 3 W/cm^2 of 3.012 μm light for 60 ms. The transition has hyperfine structure due to the nuclear spin of $I = 9/2$, which spreads the transition over ~3 GHz. Individual hyperfine transitions are not resolved in the repumping efficiency curve because of the high intensity of the repump laser and length of time over which the repumping laser is applied [39]. The 3 μm laser is tuned 1 GHz blue of the ^{87}Sr centroid, which effectively repumps transitions from the $F = 11/2$ and $F = 13/2$ $^3P_2$ levels. During the repumping stage, the 461 nm MOT is left on so that atoms returned to the ground state are recaptured and cooled. We recapture 3×10^7 atoms at temperatures of a few mK. It should be possible

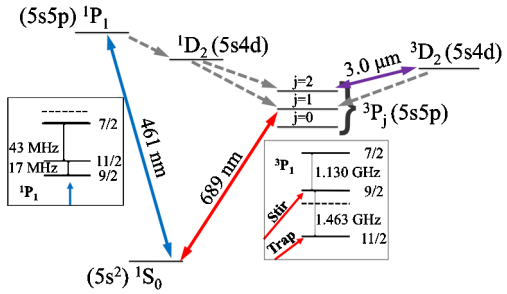

FIG. 1 (color online). Partial level diagram for laser cooling of Sr. Insets show hyperfine structure of ^{87}Sr (solid line) and isotope shifts with respect to ^{88}Sr levels (dashed line). ^{88}Sr transitions are +12 MHz detuned from the $F = 7/2$ $^1P_1$ level and −222 MHz from the $F = 9/2$ $^3P_1$ level in ^{87}Sr. For ^{87}Sr, total quantum number F is indicated.

to improve this number by modulating the repumping laser frequency to excite atoms in all of the F levels.

Following this step, the 461 nm light is extinguished and 689 nm light is applied to drive the $(5s^2)^1S_0 - (5s5p)^3P_1$ transition and create an intercombination-line MOT. Similar to [40], two frequencies are applied. The two frequencies are required because the Zeeman shift is much larger in the excited state than the ground state, causing some ground-state m_F levels to be antitrapped. A ''trap laser,'' slightly red-detuned from the $F = 9/2 \rightarrow F = 11/2$ hyperfine transition, creates a stable trapping force on average for atoms only in the presence of a ''stir laser'' red-detuned from the $F = 9/2 \rightarrow F = 9/2$ transition. The stir laser effectively randomizes the populations because the Zeeman shift for this transition is much smaller than for the trapping transition. For a strong transition, such as the $^1S_0 - {}^1P_1$ transition at 461 nm, this mixing is accomplished by a single trapping laser, but an extra laser is required for the intercombination transition because excitation is relatively slow.

To increase the initial capture rate from the 461 nm MOT, both the stir laser and trap laser are artificially broadened so that they are resonant with a wider velocity range of atoms. This is achieved by modulating the frequencies of the acousto-optic modulators generating the beams at a frequency of 30 kHz with a peak-to-peak amplitude of 660 kHz. Initial detunings are -1.2 MHz and -900 kHz from the stir and trap transitions, respectively, and intensities are 1 and 2.2 mW/cm^2. This allows trapping of up to 70% of the atoms in the intercombination-line MOT. During a 400 ms cooling time, the detuning, dither, and intensity are reduced to -90 kHz, 200 kHz, and 0.8 mW/cm^2 for the trap laser and -600 kHz, 200 kHz, and 0.9 mW/cm^2 for the stir laser. Simultaneously, the magnetic field gradient is increased from 0.2 to 1.9 G/cm. This increases spatial confinement and transfer to the optical dipole trap (ODT).

After the cooling and compression stage, an ODT, consisting of two crossed laser beams, is overlapped for 100 ms with the intercombination-line MOT with a power of 3.9 W per beam. A single beam derived from a 21 W multimode, 1.06 μm fiber laser is recycled through the chamber to form the ODT. The resulting trap has equipotentials that are nearly oblate spheroids, with the tight axis close to vertical. Within the trapping region, each beam has a waist of approximately 90 μm. During this loading time, the dither amplitudes are reduced to zero, and the intensities and detunings of the MOT beams are further reduced to approximately 3 μW/cm^2 and -30 kHz.

After the extinction of the 689 nm light, the ODT power is ramped in 30 ms to 7.5 W per beam to obtain a trap depth of 25 μK. At this point, we typically trap 3×10^6 atoms at 7 μK with a density of 2.5×10^{13} cm^{-3}. No attempt is made to spin polarize the sample. Experiments using ^{87}Sr for optical frequency standards [41] found that the intercombination-line MOT produced a sample with roughly equal populations of ground-state magnetic sublevels. For a conservative estimate of the phase-space density, we will assume equal populations in all levels. This corresponds to an average initial collision rate of 200 s^{-1} and a phase-space density of approximately 10^{-3}. As discussed below, exact determination of the polarization is not necessary to establish Fermi degeneracy of our sample and will be the topic of future studies. The sample lifetime in a static ODT, presumably limited by background gas collisions, is 30 s, which is sufficient for evaporative cooling.

After loading into the ODT, we begin forced evaporation by lowering the power in the ODT according to the formula $P = P_0/(1 + t/\tau)^\beta + P_{\text{offset}}$ with time denoted by t, $\beta = 1.4$, and $\tau = 1.5$ s. This trajectory was designed [42] without P_{offset} to yield efficient evaporation when the effect of gravity can be neglected. Gravity is a significant effect in this trap for Sr, and to avoid decreasing the trap depth too quickly at the end of the evaporation, we set $P_{\text{offset}} = 0.7$ W, which corresponds to the power at which gravity causes the trap depth to be close to zero.

For diagnostics, the ODT laser is extinguished and atoms are allowed to expand for a time of flight (TOF) between 20 and 22 ms. We then obtain absorption images using a linearly polarized laser on resonance with the $^1S_0(F = 9/2) - {}^1P_1(F = 11/2)$ transition. The intensity is $0.02I_s$, where $I_s = 42$ mW/cm^2 is the saturation intensity, and the exposure time is 35 μs. Because of the small splitting between the $F = 11/2$ and $F = 9/2$ $^1P_1$ states, the imaging beam excites transitions to both states. For a given density of atoms, the optical depth of the sample depends upon the population distribution of ground-state magnetic sublevels. Numerical solution of the optical pumping rate equations shows that atoms quickly approach a steady state distribution. For the small initial polarization in our samples, the variation in optical depth is relatively small and an average absorption cross section of $\sigma = 3\lambda^2/4\pi$, where $\lambda = 461$ nm, is accurate to within 10%. This diagnostic does not provide enough information to distinguish atoms in different magnetic sublevels.

Figure 2 shows data and fits for one-dimensional slices through the center of TOF absorption images along axes perpendicular to the imaging beam and to each other. Position has been converted to velocity assuming ballistic expansion and long expansion time. The ratio of the peak mean field interaction energy to Fermi energy is 10^{-1} for our most degenerate samples. This is relatively large for a quantum degenerate Fermi gas that is not near a collisional resonance [43] and results from the large number of occupied spin states. But the interaction energy is still small enough to neglect in our analysis.

Well outside the quantum degenerate regime, the Maxwell-Boltzmann and Fermi-Dirac momentum distributions are not distinguishable. To fit the full TOF images

accurately when the atoms are in the quantum degenerate regime, we must use the Fermi-Dirac expression for the two-dimensional momentum distribution [44]

$$\pi(p_x, p_y) = -\frac{(k_B T)^2}{2\pi m(\hbar\omega)^3}\mathrm{Li}_2[-\zeta e^{-(p_x^2+p_y^2)/2mk_BT}], \quad (1)$$

where Li_n is the polylogarithmic function of order n, $p_{x/y}$ are momenta transverse to the imaging beam, and ζ is the fugacity. The fugacity is related to the sample temperature (T) and Fermi temperature (T_F) by $\mathrm{Li}_3[-\zeta] = -1/6(T/T_F)^3$, so this method yields a measure of the quantity T/T_F without an explicit assumption on the spin polarization. Formally, the signal is a sum of distributions for each magnetic sublevel, where each may have different fugacities. Because of the weak dependence of $T_F = (6N)^{1/3}\hbar\omega/k_B$ on the number of atoms in a single quantum state (N), atoms in magnetic sublevels with the smallest T/T_F values will dominate the signal. Sublevels with significantly lower population and higher values of T/T_F will contribute less to the signal, but they do cause the extracted value of T/T_F to be an upper bound of the value for the most populated state. The weak dependence of the velocity distribution on the fugacity far from degeneracy makes extracting T/T_F difficult for hotter clouds.

When fitting the TOF images using Maxwell-Boltzmann statistics, the entire image is fit with a Gaussian. This fit determines the number of atoms [44] and the size of the cloud s, where the density is $n \propto \exp(-r^2/2s^2)$. To determine the temperature, the high velocity wings of the distribution, consisting only of atoms at radii greater than s, are fit to a Maxwell-Boltzmann velocity distribution. The deviation of the full distribution from a Gaussian is never large, and the high velocity wings are less sensitive to the changes in the shape of the distribution as the atoms become degenerate [13]. Numerical simulations of velocity distributions show that this method of thermometry is accurate to within 10% for $T/T_F > 0.4$ and becomes less accurate for more degenerate samples. The temperature is always overestimated. Combining this measure of temperature with knowledge of the ODT oscillation frequencies provides a measurement of T/T_F. Here we assume equal population of the magnetic substates, which makes this an upper bound on T/T_F for the magnetic sublevel with the largest population.

After 4.1 s of forced evaporation, the sample has reached $T/T_F = 1$ as measured by both methods of fitting, although the uncertainty is large for the Fermi-Dirac fitting. Both the Maxwell-Boltzmann and the Fermi-Dirac fitting describe the data well (Fig. 2). After 10.5 s the Maxwell-Boltzman distribution overestimates the number of atoms with small velocity in the central region of the cloud. However, the Fermi-Dirac fitting yields an excellent description of the data with a fugacity corresponding to $T/T_F = 0.26^{+0.05}_{-0.06}$. The non-Gaussian character of the distribution is a clear signature of the onset of quantum degeneracy and the limiting of occupancy of lower energy levels due to the Pauli exclusion principle.

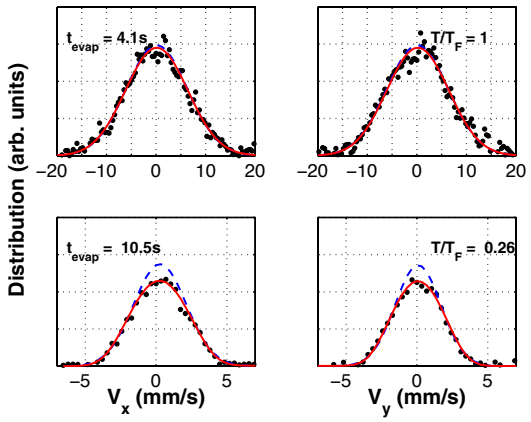

FIG. 2 (color online). Velocity distributions along axes perpendicular to imaging beam. At $t = 4.1$ s of forced evaporation (top), the data are fit well with a Maxwell-Boltzmann (dotted line) or Fermi-Dirac distribution (solid line). At 10.5 s, classical-particle statistics overestimates the population at low velocities, while Fermi-Dirac statistics for $T/T_F = 0.26$ accurately fit the data.

Figure 3 shows the evolution of T/T_F during the forced evaporation, determined by the two methods described above. The temperature starts at 4 μK and reaches 30 nK after 10 s of evaporation. We achieve a limiting value

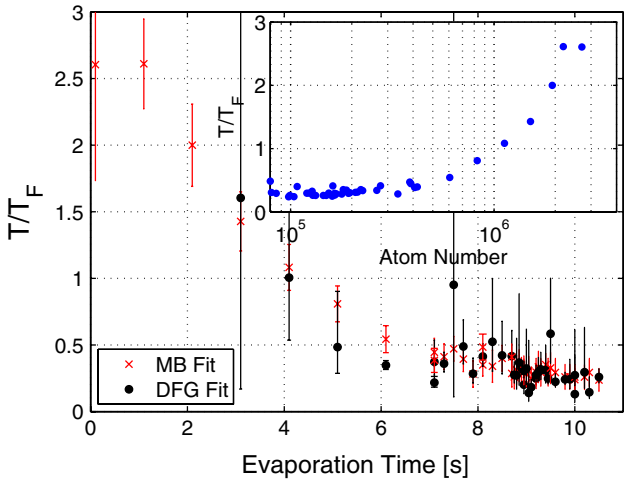

FIG. 3 (color online). Evaporation of ^{87}Sr. T/T_F is obtained from either a fit of the velocity distribution to a single Fermi-Dirac distribution (DFG) or a determination of number and temperature from fits to a Maxwell-Boltzmann distribution (MB) and knowledge of the trap oscillation frequencies. Inset: Variation of T/T_F with atom number. T/T_F is calculated using the latter of the two methods. Evaporation is efficient early in the trajectory and T/T_F decreases significantly as atoms are lost. But T/T_F plateaus near the limit of about ~0.25 as the evaporation efficiency decreases.

$T/T_F = 0.25$ with this trajectory, and the agreement of the two methods of determining this quantity indicate that the assumption of equal ground-state sublevel populations is a reasonable approximation for this data.

Figure 3 (inset) provides information on why lower values of T/T_F are not observed. Evaporation initially proceeds efficiently through collisions of ^{87}Sr atoms in different magnetic sublevels. T/T_F drops by a factor of almost 10 for a loss of about a factor of 10 in atom number. At values below $T/T_F = 0.5$, the evaporation becomes inefficient. This signature has been seen in evaporation of ^{40}K [13] and ^{173}Yb [16] and ascribed to Pauli blocking of collisions when atoms have reached the quantum degenerate regime.

We have described the creation of a quantum degenerate Fermi gas of ^{87}Sr. This result opens possibilities of future studies involving quantum degenerate Bose-Fermi mixtures [27,28], quantum degenerate Fermi gases with large ground-state degeneracy [18–20], and quantum computing with alkaline-earth metal atoms [22–24].

This research was supported by the Welch Foundation (C-1579), National Science Foundation (PHY-0855642), and the Keck Foundation.

[1] S. Giorgini, L. Pitaevskii, and S. Stringari, Rev. Mod. Phys. **80**, 1215 (2008).
[2] C. Chin, R. Grimm, P. Julienne, and E. Tiesinga, Rev. Mod. Phys. **82**, 1225 (2010).
[3] I. Bloch, J. Dalibard, and W. Zwerger, Rev. Mod. Phys. **80**, 885 (2008).
[4] G. B. Partridge, W. Li, R. I. Kamar, Y.-a. Liao, and R. G. Hulet, Science **311**, 503 (2006).
[5] M. W. Zwierlein, A. Schirotzek, C. H. Schunck, and W. Ketterle, Science **311**, 492 (2006).
[6] K. M. O'Hara, S. L. Hemmer, M. E. Gehm, S. R. Granade, and J. E. Thomas, Science **298**, 2179 (2002).
[7] C. Chin *et al.*, Science **305**, 1128 (2004).
[8] M. W. Zwierlein, J. R. Abo-Shaeer, A. Schirotzek, C. H. Schunck, and W. Ketterle, Nature (London) **435**, 1047 (2005).
[9] M. Köhl, H. Moritz, T. Stöferle, K. Günter, and T. Esslinger, Phys. Rev. Lett. **94**, 080403 (2005).
[10] R. Jordens, N. Strohmaier, K. Gunter, H. Moritz, and T. Esslinger, Nature (London) **455**, 204 (2008).
[11] W. Hofstetter, J. I. Cirac, P. Zoller, E. Demler, and M. D. Lukin, Phys. Rev. Lett. **89**, 220407 (2002).
[12] L.-M. Duan, E. Demler, and M. D. Lukin, Phys. Rev. Lett. **91**, 090402 (2003).
[13] B. DeMarco and D. S. Jin, Science **285**, 1703 (1999).
[14] A. G. Truscott, K. E. Strecker, W. I. McAlexander, G. B. Partridge, and R. G. Hulet, Science **291**, 2570 (2001).
[15] J. M. McNamara, T. Jeltes, A. S. Tychkov, W. Hogervorst, and W. Vassen, Phys. Rev. Lett. **97**, 080404 (2006).
[16] T. Fukuhara, Y. Takasu, M. Kumakura, and Y. Takahashi, Phys. Rev. Lett. **98**, 030401 (2007).
[17] Congjun Wu, Mod. Phys. Lett. B **20**, 1707 (2006).
[18] A. V. Gorshkov, M. Hermele, V. Gurarie, C. Xu, P. S. Julienne, J. Ye, P. Zoller, E. Demler, M. D. Lukin, and A. M. Rey, Nature Phys. **6**, 289 (2010).
[19] M. Hermele, V. Gurarie, and A. M. Rey, Phys. Rev. Lett. **103**, 135301 (2009).
[20] M. A. Cazalilla, A. F. Ho, and M. Ueda, New J. Phys. **11**, 103033 (2009).
[21] J. Ye, H. J. Kimble, and H. Katori, Science **320**, 1734 (2008).
[22] A. J. Daley, M. M. Boyd, J. Ye, and P. Zoller, Phys. Rev. Lett. **101**, 170504 (2008).
[23] A. V. Gorshkov, A. M. Rey, A. J. Daley, M. M. Boyd, J. Ye, P. Zoller, and M. D. Lukin, Phys. Rev. Lett. **102**, 110503 (2009).
[24] I. Reichenbach, P. S. Julienne, and I. H. Deutsch, Phys. Rev. A **80**, 020701(R) (2009).
[25] M. Foss-Feig, M. Hermele, and A. M. Rey, Phys. Rev. A **81**, 051603(R) (2010).
[26] S. Stellmer, M. K. Tey, B. Huang, R. Grimm, and F. Schreck, Phys. Rev. Lett. **103**, 200401 (2009).
[27] Y. N. Martinez de Escobar, P. G. Mickelson, M. Yan, B. J. DeSalvo, S. B. Nagel, and T. C. Killian, Phys. Rev. Lett. **103**, 200402 (2009).
[28] P. G. Mickelson, Y. N. Martinez de Escobar, M. Yan, B. J. DeSalvo, and T. C. Killian, Phys. Rev. A **81**, 051601(R) (2010).
[29] A. Imambekov and E. Demler, Phys. Rev. A **73**, 021602 (R) (2006).
[30] R. Ciurylo, E. Tiesinga, and P. S. Julienne, Phys. Rev. A **71**, 030701(R) (2005).
[31] K. Enomoto, K. Kasa, M. Kitagawa, and Y. Takahashi, Phys. Rev. Lett. **101**, 203201 (2008).
[32] Y. N. Martinez de Escobar, P. G. Mickelson, P. Pellegrini, S. B. Nagel, A. Traverso, M. Yan, R. Côté, and T. C. Killian, Phys. Rev. A **78**, 062708 (2008).
[33] P. G. Mickelson, Y. N. Martinez de Escobar, P. Anzel, B. J. DeSalvo, S. B. Nagel, A. J. Traverso, M. Yan, and T. C. Killian, J. Phys. B **42**, 235001 (2009).
[34] M. Yasuda and H. Katori, Phys. Rev. Lett. **92**, 153004 (2004).
[35] G. Ferrari, R. E. Drullinger, N. Poli, F. Sorrentino, and G. M. Tino, Phys. Rev. A **73**, 023408 (2006).
[36] S. B. Nagel, C. E. Simien, S. Laha, P. Gupta, V. S. Ashoka, and T. C. Killian, Phys. Rev. A **67**, 011401(R) (2003).
[37] X. Xu, T. H. Loftus, J. L. Hall, A. Gallagher, and J. Ye, J. Opt. Soc. Am. B **20**, 968 (2003).
[38] N. Poli, R. E. Drullinger, G. Ferrari, J. Léonard, F. Sorrentino, and G. M. Tino, Phys. Rev. A **71**, 061403(R) (2005).
[39] P. G. Mickelson, Ph.D. thesis, Rice University, 2010.
[40] T. Mukaiyama, H. Katori, T. Ido, Y. Li, and M. Kuwata-Gonokami, Phys. Rev. Lett. **90**, 113002 (2003).
[41] A. D. Ludlow, Ph.D. thesis, University of Colorado at Boulder, 2008.
[42] K. M. O'Hara, M. E. Gehm, S. R. Granade, and J. E. Thomas, Phys. Rev. A **64**, 051403(R) (2001).
[43] C. J. Pethick and H. Smith, *Bose-Einstein Condensation in Dilute Gases* (Cambridge University Press, Cambridge, England, 2008).
[44] B. DeMarco, Ph.D. thesis, University of Colorado, 2001.

INTRODUCTION

Before the completion of this work, all experiments performed in the Killian lab had involved either thermal atoms or Bose-Einstein Condensates (BEC). As such, the analysis protocol for these gases was well established in the lab, but the tools necessary to fit a quantum degenerate Fermi gas had yet to be built. The development of these tools will be the subject of this appendix.

The importance of a good fitting routine when working with a degenerate Fermi gas can not be overstated. The phase transition from a gas of thermal bosons to a BEC as viewed in time-of-flight images is dramatic. Below the critical temperature, atoms pour into the ground state of the confining harmonic potential and create a sharp spike at low momenta. Even in the absence of any fitting routine, the bimodal structure is clear to the naked eye (or at least a naked eye aided by a high resolution optical imaging system), and provides unambiguous proof of the onset of quantum degeneracy.

In stark contrast, the onset of degeneracy in the case of fermions is subtle. There is no phase transition, only a smooth crossover to a regime where quantum statistics become important. The Pauli exclusion principle forbids identical fermions from occupying the same energy level in the confining potential, so at zero temperature, the atoms fill levels up to the Fermi energy. Finite temperature excites a fraction of these atoms to higher energy levels. For the temperatures reached in our experiment, the momentum distribution is never far from what one expects for a gas of thermal atoms. Therefore careful fitting is necessary to tease out the effects of degeneracy as seen in time-of-flight images.

This appendix is separated into two parts and will be written with an eye for an experimentalist working with ^{87}Sr . The first will contain theory and details of the analysis of a non-interacting degenerate Fermi gas. The second will describe how interatomic interactions can be taken into account and their effect on the momentum distribution.

THE NON-INTERACTING FERMI GAS

Theory

The theory of a non-interacting degenerate Fermi gas is easily worked out using elementary statistical mechanics. This section will describe the basic calculation of relevant

quantities measured in our experiment in terms of parameters familiar to anyone working with ultra-cold atoms.

We begin with the single particle Hamiltonian for an atom trapped in a harmonic potential

$$H = \frac{1}{2m}(p_x^2 + p_y^2 + p_z^2) + \frac{m\omega^2}{2}(x^2 + y^2 + z^2) \tag{1}$$

where m is the mass of the atom, p_i is the atom's momentum in the i direction and ω is the geometric mean of the trapping frequencies.

For the case of our experiment where $k_B T >> \hbar\omega$, the density of states can be written as [1]

$$g(\epsilon) = \frac{\epsilon^2}{2(\hbar\omega)^3} \tag{2}$$

where ϵ is the energy of a particle. Of course, the familiar Fermi-Dirac distribution function is

$$F(\epsilon) = \frac{1}{\frac{1}{\zeta}e^{\frac{\epsilon}{k_B T}} + 1} \tag{3}$$

where ζ is the fugacity.

These quantities can be related to the number of atoms of a given spin species, N_σ via

$$N_\sigma = \int_0^\infty g(\epsilon)F(\epsilon)\,\mathrm{d}\epsilon. \tag{4}$$

At zero temperature, $F(\epsilon) = 1$ for energies less than the Fermi energy, E_F, and zero otherwise. The integral is then trivial, and allows us to calculate the Fermi temperature in terms of easily experimentally accessible quantities

$$\boxed{T_F = \frac{E_F}{k_B} = \frac{\hbar\omega}{k_B}(6N_\sigma)^{1/3}.} \tag{5}$$

At finite temperature, the above integral is more challenging, however an analytic solution exists. Using your favorite table of integrals, one finds [2]

$$N_\sigma = -(\frac{k_B T}{\hbar\omega})^3 Li_3(-\zeta) \tag{6}$$

Also, one can calculate the total energy of the gas

$$U = \int_0^\infty \epsilon g(\epsilon)F(\epsilon)\,\mathrm{d}\epsilon = -3\frac{(k_B T)^4}{(\hbar\omega)^3}Li_4(-\zeta) \tag{7}$$

where Li_n is the n^{th} order Poly-Logarithmic function. Combining these results and inserting the definition of T_F one finds

$$\boxed{Li_3(-\zeta) = -\frac{1}{6(T/T_F)^3}} \tag{8}$$

This relation is extremelly useful and allows us to relate the fugacity directly to the degeneracy parameter T/T_F. The advantages of this will be discussed further in the next section.

We now turn our attention to the calculation of the momentum distribution. This can be done in the Thomas-Fermi approximation, a semi-classical approximation valid for our experiment [1]. The crux of the approximation is that although the potential energy has a position dependence, the gas behaves locally as a homogeneous system. Within this approximation, one can write down the number density in phase space as

$$w(\vec{r},\vec{p}) = \frac{1}{(2\pi\hbar)^3}\frac{1}{\frac{1}{\zeta}e^{\frac{H(\vec{r},\vec{p})}{k_BT}}+1} \tag{9}$$

The momentum space distribution can be obtained by integrating over $\vec{r}$

$$\pi(p_x,p_y,p_z) = -\frac{1}{(2\pi)^{3/2}\hbar}(\frac{k_BT}{m\omega})^{3/2}Li_{3/2}(-\zeta e^{p_x^2+p_y^2+p_z^2/2m}) \tag{10}$$

In our experiment, we probe the momentum distribution via absorption imaging of time-of-flight images. In the absence of interactions, the expansion of the cloud is purely ballistic, and the real space distribution of atoms in the image reflects the momentum distribution at the moment the atoms were released from the trap. The 2D image recorded is therefore related to the momentum distribution integrated over the z direction and multiplied by the optical cross section of the resonant light, $\sigma_{opt} = 3\lambda^2/2\pi$ where λ is the wavelength of the transition used for imaging. Writing this in terms of useful quantities, one obtains the optical depth after an expansion time, t

$$\boxed{OD(x,y) = OD_{peak}\frac{Li_2(-\zeta e^{-x^2/2\sigma_{TF,x}^2}e^{-y^2/2\sigma_{TF,y}^2})}{Li_2(-\zeta)}} \tag{11}$$

where OD_{peak} is the peak optical depth and $\sigma_{TF,i} = \sqrt{\frac{k_BT}{m\omega_i^2}(1+\omega t^2)}$.

For reference, doing the same calculation for a gas of classical particles yields the following expression

$$OD(x,y) = OD_{peak} e^{-x^2/2\sigma_x^2} e^{-y^2/2\sigma_y^2} \tag{12}$$

Fitting the Cloud

With the theoretical background in place, we can get into the business of fitting the images obtained in the experiment. As mentioned before, at the time the experiment was performed, the tools were already in place to fit a time-of-flight image of a thermal gas of atoms, but not a degenerate Fermi gas. The routine described here is a consequence of this history, and allowed us to first see hints that there was a deviation in the momentum distribution from what one expects for classical particles as well as calculate relevant quantities of interest while at the same time having checks in place to make sure the results of the fitting routine were reasonable. So while this scheme of fitting the cloud may seem overly complicated, there is a method to the madness.

First, the entire cloud is fit to equation 12, as expected for a trapped gas of classical particles. Even for our most degenerate samples, $T/T_F = .26$, the measured distribution does not seem far from gaussian. From these fits, peak optical depth and widths σ_x and σ_y are extracted. At high temperatures, one trusts these parameters to calculate number and temperature respectively. However caution should always be used for imaging ^{87}Sr on the $^1S_0 - ^1P_1$ transition. Due to optical pumping during the imaging time, the effective optical cross section is reduced by a factor of 2, and using the equation 12 will cause an underestimate of the atom number by a factor of 2. A detailed discussion of this effect can be found in [3]. At low temperatures, the fit values fail to calculate number and temperature due to an incorrect assumption in the distribution. However the calculated number is more accurate than the calculated temperature due to the way in which the momentum distribution changes. So we will use the number of atoms from this stage of fitting, but the fit widths will only be used as an input for the next stage.

In the second stage, only the high momentum wings of the distribution are fit to a gaussian distribution. Programatically, this is done by giving no weight to points within a square centered on the cloud with sides of length 2σ where $\sigma = \frac{\sigma_x + \sigma_y}{2}$ as the data is passed to the fitting routine in Matlab.

The reason for this second stage is as follows: for a degenerate Fermi gas, information

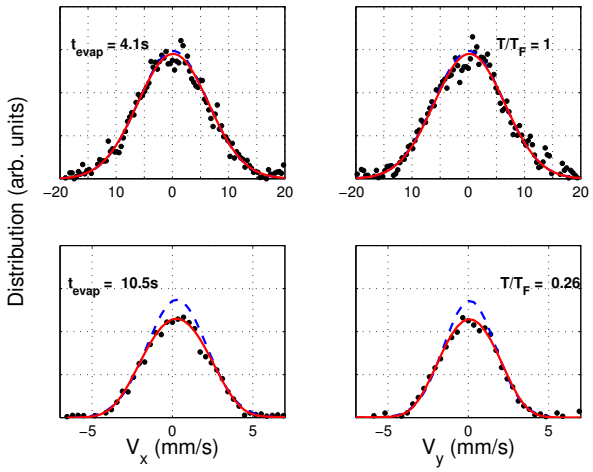

FIG. 1. Velocity distributions along axes perpendicular to imaging beam. At $t = 4.1\,\mathrm{s}$ of forced evaporation (top), the data is fit well with a Maxwell-Boltzmann(- -) or Fermi-Dirac distribution(–). At 10.5 s, the gaussian fit to the wings overestimates the population at low velocities.

about temperature from the momentum distribution is encoded in atoms excited above the Fermi momentum. At or near this value, the distribution is much closer to the classical distribution than the low momentum atoms deep in the Fermi sea. Therefore, fitting just the wings of the distribution yields a more accurate temperature measurement for degenerate samples. For thermal clouds, this method will still accurately fit the entire cloud. As shown in figure 2 of the paper, reprinted here for ease of reference, this is true even for clouds at the Fermi temperature. However, for deeply degenerate samples, stage 2 over-estimates the density in the center of the cloud.

We have tested the results of stage 1 and stage 2 on known distributions, and find σ_x and σ_y from stage 2 fits results in a more accurate measure of the temperature for degenerate samples, as shown in figure 2, and makes mainly small errors for the majority of our evaporation trajectory. At this point, we can get a reasonable estimate of the degeneracy parameter, T/T_F by combining our measured T with T_F calculated using equation 5 using the number calculated in stage 1 and the temperature calculated in stage 2. It should be noted here that uncertainty in the calculation of the degeneracy parameter is dominated by uncertainty in the knowledge of the trapping potential. Ideally, one would measure the trapping frequencies at each point of interest in the evaporation trajectory, but this is tedious work and undesirable. Instead, we rely on modeling of the confining potential as described

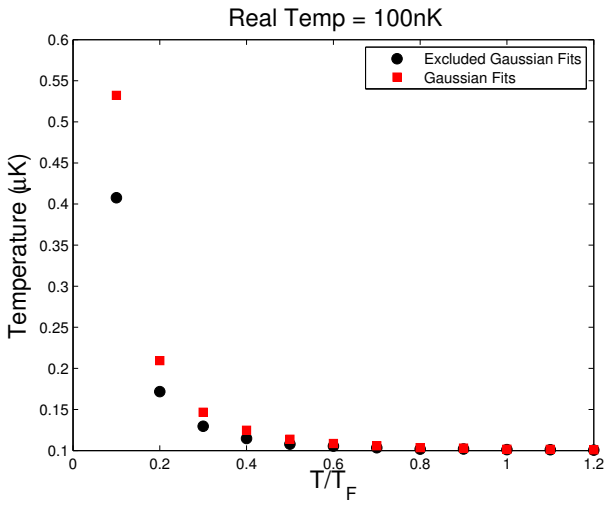

FIG. 2. Comparison of the results of stage 1 and 2 fitting on a known distribution of a degenerate Fermi gas with varying T/T_F

in [4]. This does a good enough job for us to get reasonable estimates as to how degenerate the sample becomes.

At this point in the analysis, it should be clear whether there is a deviation from classical statistics or not. However, the methods used so far are not perfectly appropriate for fitting degenerate samples. As a final stage of the analysis, we fit the atoms to equation 11, the Thomas-Fermi profile. The importance of equation 8 is now apparent. Using fugacity as a fitting parameter, one can extract T/T_F directly from the fits with no knowledge of the the trapping potential. However, it should be noted that fitting hotter clouds with this distribution is much less reliable due to the weak dependence on the fugacity.

As with any non-linear fitting routine, it is essential to make a reasonable guess for the parameters to be fit in order for the routine to converge. While this is easy for most of the fitting parameters, the fugacity poses a challenge. The estimate of T/T_F from stage 2 is good, but one needs to calculate the fugacity corresponding to that value. Unfortunately, the Poly-Logarithmic functions are not native to MatLab. There is a package one can use, named dilog.m, which will accurately calculate Poly-Logarithmic functions of order 2. This is useful for defining the fit function that we need, but to relate fugacity and T/T_F we need order 3 Poly-Logarithmic functions. Fortunately, these can be calculated in Mathematica.

For the purposes of this paper and the results obtained, an initial guess of a fugacity of 1 was able to fit most of the data, and manually changing the initial guess worked to

fit the most degenerate samples. The lack of automation in this step is undesirable, but allows us to skip one step of passing data back and forth between MatLab and Mathematica. The fit fugacities from $imagefit\_binaryread\_GaussianExtraction\_FitFermions.m$ are then written to an output file, $filenamefugacity.txt$, which can be read into a Mathematica notebook, $FugacitytoReducedTempWithError.nb$, where T/T_F associated with the fugacity is calculated, and written to an output file, $filenameReducedTemp.txt$. Delayimageanalyze.m, can then read in this data and plot it. Copies of these programs can be found in $\backslash\backslash Magnesium\backslash Analysis\backslash\ Degenerate\_Fermi\_Gas\_Analysis$.

Clearly, this kludged solution is far from ideal. However, MatLab is much better suited for the application of fitting images, and only Mathematica can calculate the Poly-Logarithmic functions we need. For the time being, it will have to do until MatLab is able to handle Poly-Logarithmic functions of arbitrary order.

This complicated routine in the end provides a reliable way of fitting clouds of degenerate Fermi gasses under the assumption that the interatomic interactions are negligible. The three stages also allow comparison to give insight into what is going on. Comparing stage 1 and stage 2 allows one to see the first signs of degeneracy. Comparing stage 2 and stage 3 allows one to check that the fit value of the degeneracy parameter is reasonable based on your knowledge of the trap. However, the result of stage 3 is independent of the trap and yields the most trustworthy results for degenerate Fermi gasses.

THE INTERACTING FERMI GAS

The interacting degenerate Fermi gas is a rich topic in physics. Here our simple statistical mechanics fails, and ones needs to unleash the power of many-body techniques to capture the essential physics at play. A proper treatment of the problem is beyond the scope of this work, but instead I will try to answer one simple question: How much does the presence of interactions change the momentum distribution we probe? The answer to the question is not enough to worry about it.

If one looks in the literature, for a degenerate Fermi gas not near a collisional resonance, most experimentalists do not worry about the effect of interactions. However, these experiments, with few exceptions, are performed with alkali atoms. At most, a two-component degenerate Fermi gas is obtainable, but with Sr we create a 10 component degenerate Fermi

gas. It is this fact that makes interactions stronger in the degenerate Fermi gas we create.

Before diving into the calculation, it is always useful to do a "back of the envelope" check to get some sense of how much of an effect one expects. The relevant quantity we are interested is the ratio of the interaction energy to the Fermi energy. The interaction energy is defined as follows

$$U_{int} = \frac{4\pi\hbar^2 a}{m} n_\sigma \tag{13}$$

where n_σ is the density of a single spin state, and a is the scattering length. In terms of the Fermi energy, one can write the density as

$$n_\sigma = \frac{N_\sigma}{\frac{4}{3}\pi R_F^3} = \frac{3\hbar^2 a\omega^3 m^{1/2}}{(2E_F)^{3/2}} N_\sigma \tag{14}$$

Taking the ratio, and using the definition of the Fermi energy, equation 5, one obtains

$$\frac{U_{int}}{E_F} \approx 0.23(\frac{m\omega a^2}{\hbar})^{1/2} N_\sigma^{1/6}. \tag{15}$$

Plugging in reasonable numbers for our experiment, $a = 97a_0$, $\omega/2\pi = 100$ Hz, $N_\sigma = 10^4$, we find $U_{int}/E_F = .005$. This effect is small, but only accounts for interactions between one species and another. There are ten species present, so we must multiply the result by 9 and the ratio grows to 4.5%. While still small, it is hard to argue that it is negligible. As a result, we will treat the interactions more completely and compare the results to the non-interacting case.

To account for the effect of interactions of the momentum distribution, we will follow the same prescription from the previous section. In the Thomas-Fermi approximation, equation 9 still holds, but we will add in an interaction term to the Hamiltonian, again with the assumption of equal spin state populations.

$$H_{Int} = H_0 + 9\frac{4\pi\hbar^2 a}{m}(n_\sigma(r) - n_\sigma(0)) \tag{16}$$

Note the factor of 9 accounts for different components of the gas.

To give an upper bound on the effect, we use the zero temperature density distribution for a degenerate Fermi gas [5]

$$n_\sigma(r) = \frac{8}{\pi^2}\frac{N_\sigma}{R_F^3}(1 - r^2/R_F^2)^{3/2}\Theta(1 - r^2/R_F^2) \tag{17}$$

Using this, and integrating over $\vec{r}$ as before yields an integral without an analytic solution. However, Mathematica is well suited to handle this and using NIntegrate one can numerically

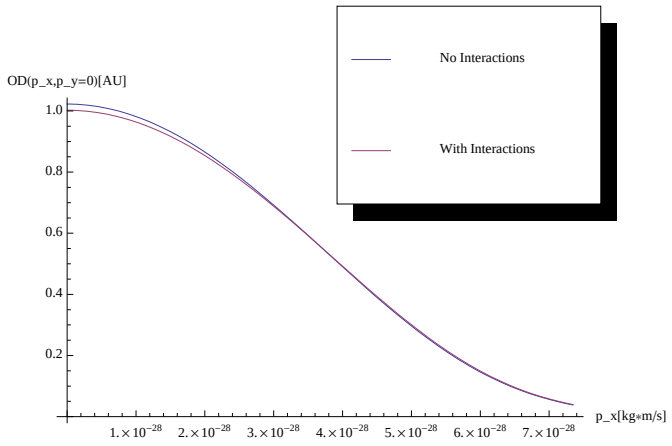

FIG. 3. Comparison of the expected optical depth with and without interactions included in the Hamiltonian

evaluate the integral. Then setting the scattering length to zero, one can compare the results of the interacting and non-interacting case, as shown in figure 3.

As seen in the plot, the effect of interactions is very small and does not contribute significantly to the momentum distribution. From this we conclude that the effect of interactions is not significant and can safely be ignored.

CODE

For reference, this is the code of the programs mentioned in the text in it's current form. The first program is $imagefit\_binaryread\_GaussianExtraction\_FitFermions.m$, a Matlab program used for the majority of the fitting. Note, this program has a lot of functionality and also works for fitting BEC and thermal atoms. However, the best program for fitting BEC resides elsewhere, and should eventually be merged with this program.

```
%This program is designed to read several datafiles and corresponding background
%files,  plots these normalized data sets on the same graph.
close all
clear all
clc

%% Directory and Files
%directory=char('\\Magnesium\neutraldata\Sr87\2010.01.19\'); %Primary directory where all output files are saved%
%directory=char('C:\data\Sr87\2010.01.19\'); %for Magnesium PC
%batchdirectory=char('\\Strontium.rice.edu\strontium86\GROUP PAPERS\2010MixedGasDegeneracy\PureBEC\'); %Primary directory where all output files are saved%
directory=char('/Volumes/neutraldata/Sr87/2011.02.15/'); %Mac Syntax
batchdirectory=char('/Volumes/neutraldata/Sr87/2011.02.15/');
batchhead=char('Files_20110215');
basenamelist=char(strcat(deblank(batchdirectory),deblank(batchhead),deblank('.txt')))

[basenamevector timevector dummy01 dummy02 droptime roicol1 roicol2 roirow1]=...
    textread(basenamelist, '%s%f%f%f%f%f%f%f','commentstyle','matlab'); %read in data file for atoms
temp=size(basenamevector);
numberofbasenames=temp(1);%max of ten for figure 200 to work


%% Initialization and Options
mass=87*1.672*10^(-27); %strontium mass
kBoltz=1.38*10^(-23);
templow=1*10^(-6);
hbar=1.05*10^(-34);
timefactor=10^3; %scales time from ms to s
lambda=461*10^(-9);
detuning=-0.72*10^6; % image beam detuning in Hz
naturalwidth=32*10^6; %ion fwhm in Hz
textsize=16;
axisfontsize=14;

%All frequencies here ONLY go into alldata(k,1); frequencies.
dzeeman=-251.43;    %constant overnight
dPAS=-250;          %constant overnight
dsatabs=-41;        %constant overnight
dshift=0;           % 0 1st run, 260.5 for 2nd run
catseyesign=0;      %depends on peak

% HalfBEC=3;  %number of pixels of half the size of BEC
% BECweight=10^6;
% xoffsetH=51; %Blue
% yoffsetH=48;    %Red

%SPECIFICS OF FITTING ROUTINES
smoothplotsize=2; %number of points to average for plotting
guesssmoothsize=1; %This variable is used in fitting, and also smoothes out the linear density cuts
sampleinterleave=1; %for big matrices form a matrix using only one point for every sampleinterleave points in a row/column, Always=1 for final analysis and error calc
%%-------------------------------------------------------------------------------%%
%PIXEL SIZE CHANGED FROM 10.4e-6 TO 9.2e-6 ON October 12, 2009 FOR BEC DATA
%ANALYSIS FROM DATASET WITH TIMESTAMP 2539 ONWARD (FILES FOUND IN ODT\2009.10.10 DATA FOLDER)
%SEE PG. 19YELLOW OF NEUTRAL NOTEBOOK H FOR MORE DETAILS
pixelsize=0.00000857;
pixelconv=8.57; %converts pixels to microns %1.05*10^1;
%%-------------------------------------------------------------------------------%%

softwarebinsize=1; %Number of pixels square to bin (i.e. area of bin, in # of pixels = softwarebinsize*softwarebinsize
CCDbinning=1; %number of pixels binned when first recording data
prebinning=softwarebinsize*CCDbinning*sampleinterleave;
sizefactor=pixelsize*prebinning;
cutborders=0; %number of column and rows to trim on either side of the data matrix; same as roi parameters, 'cept inverse way of doing it.
limit=5; %number of column and rows at the corners to find the uncertainties/noise
scalebaselevel=0;% 1 to scale the base level of atom and background image, 0 for not
numberforscale=5;% number of column and rows to scale the base level of atom and background images
matrixsize=[1280 1024]; %matrix size for binary reading.


singlefit=0;        %0 to plot entire basename list, 1 to plot just the specified basename
plotfits=1;         %1 to plot fits/data, 0 to not plot fits; 2D Gaussian
plotevolution=1;    %1 to show surface plots for each data point on one plot
plotdensity=1;      %1 to show plots of density as function of x position (slices along y); helps to spot BEC,fig 2000,3000
standalone=1;       %1 to plot data by itself (not fits, residuals, etc...); 0 to produce normal plots
fit=1;              %1 to fit, 0 just to plot; 2D Gaussian
fit600=0;           %1 to plot fig(600+k)
becweight=0;        % 1 will set small weights to a region selected with HalfBEC, xoffset, H, etc
becfit=0;           %1 will use TF function to fit BEC...only on first passs...not on residuals
dfgfit=1;           %1 will use Fermi-Dirac function to fit cloud
fitresiduals=0;
plotresiduallinedensity=0;  % one to show line cuts of residual density and fit...must have fitresiduals=1
binaryread=1;       %0 if the files are ASCII; 1 if the files are binary
save2places=0;      %1 means we save output files to both primary and secondary directories for backup purposes

COLORS=[0 0 0;1 0 0;0 1 0; 0 0 1; .25 .25 .25; 0 .75 0; 0 .75 .75; .5 0 .5; .75 0 .75;0 0 1;0 0 0;1 0 .25;0 .75 0; .5 0 .5; 0 1 0; 0 .75 0; 0 .75 .75; .5 0 .5; .75 0 .75;0 0 1;0 0 0;1 0 .25;0 .75 0; .5 0 .5; 0 1 0; 0 .75 0; 0 .75 .75; .5 0 .5; .75 0 .75;0 0 1;0 0 0;1 0 .25;0 .75 0; .5 0 .5; 0 1 0; 0 .75 0; 0 .75 .75; .5 0 .5; .75 0 .75;0 0 1];
MARKERS={'o','s','^','d','v','p','<','h','>','-x','-*','-+','-.','-o','-s','-^','-d','-v','-p','-<','-h','->','-x','-*','-+','-.','-o','-s','-^','-d','-v','-p','-<','-h','->','-x','-*','-+','-.'};

r1=0;       %r1 and r2 used for annular ring stuff in plasma experiment.
r2=50;



%% Loop Through Basenames
filecounter=0;   %this will keep track of all the files analyzed in all the batches
for basenamenumber=1:numberofbasenames
   filecounter=filecounter+1

        if droptime(filecounter)==11 %33ms dropping time
         cloudycenter=ones(1,100)*795;   %row center
          cloudxcenter=ones(1,100)*590;  % column center
          windowradius=15;
```

```
        BECweight=10^6;
%           HalfBEC=5;  %number of pixels of half the size of BEC
%           xoffsetH=51; %Blue
%           yoffsetH=49;    %Red


        elseif droptime(filecounter)==20 %20ms dropping time
      % % %%% For 20ms drop
        cloudycenter=ones(1,20)*610;   %row center
        cloudxcenter=ones(1,20)*580;  % column center
        %for p1
         windowradius=50;
        %for p2
         %windowradius=80;
        %for p3
         %windowradius=80;
        %for p4
         %windowradius=80;
        %for p5
         %windowradius=80;
        %for p6
         %windowradius=35;
        %for p7
         %windowradius=30;
        %for p8
         %windowradius=25;
        %for p9
         %windowradius=20;


        BECweight=10^6;
         %HalfBEC=ones(1,100).*11;  %number of pixels of half the size of BEC
         %xoffsetH=ones(1,100).*109; %Blue
         %yoffsetH=ones(1,100).*125;    %Red

        elseif droptime(filecounter)==22 %22ms dropping time
         cloudycenter=ones(1,100)*560;   %row center
         cloudxcenter=ones(1,100)*585;  % column center
         windowradius=20;

        BECweight=10^6;
         %HalfBEC=ones(1,50).*2;  %number of pixels of half the size of BEC
         %xoffsetH=ones(1,50).*43; %Blue
         %yoffsetH=ones(1,50).*45;    %Red

        elseif droptime(filecounter)==16 %16ms dropping time
      % % %%% For 16ms drop
        cloudycenter=ones(1,20)*655;   %row center
        cloudxcenter=ones(1,20)*585;  % column center
        windowradius=48;

        BECweight=10^6;
%           HalfBEC=5;  %number of pixels of half the size of BEC
%           xoffsetH=48; %Blue
%           yoffsetH=69;    %Red
        end

    gaussbin=1;
    cloudycenter=cloudycenter./gaussbin;
    cloudxcenter=cloudxcenter./gaussbin;

     roirowstop=cloudycenter+windowradius;
     roirowstart=cloudycenter-windowradius;
     roicolstart=cloudxcenter-windowradius;
     roicolstop=cloudxcenter+windowradius;
     variableROI=1;

    clear basename;
    clear file ;
    clear imagevco;
    clear vcovoltage;
    clear motdetuning;
    clear d;
    clear delay;
    clear ODTVoltage
    clear dummy
    clear dummy2
    clear dummy3
    clear delaytime

    if singlefit==0;
        basename=char(basenamevector(basenamenumber));
       batchfile=char(strcat(deblank(directory),deblank(basename),deblank('.batch')));    %array of batchfile names

      [file imagevco vcovoltage motdetuning ODTVoltage delay dummy dummy2 dummy3 delaytime samplehold wavemeter]=textread(batchfile, '%q%f%f%s%f%f%f%f%f%f%f
%s','commentstyle','matlab'); %read in all files, no limit
       temp=size(imagevco);

       basenamesize=temp(1);%automatically determine how many files in batch associated with this basename
       q=[basenamesize];% array of number of data files in each batch

       batchdelay(1:q,basenamenumber)=delay(1:q); %make array of all delays
       batchvcovoltage(1:q,basenamenumber)=vcovoltage(1:q); %make array of all vco voltages
       batchimagevco(1:q,basenamenumber)=imagevco(1:q); %make array of all image vco voltages

       %%Set exclusion region based on gaussian fits
       if becweight==1 && becfit==0
          gaussianfitfile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('2Dfitparams.txt')));  %simple gaussian fits
          [f1 f2 f3 f4 f5 f6 positionx positiony f9 peakOD sigmax sigmay f13 f14 f15 f16 f17 f18 sigmaxerror sigmayerror f21 f22 f23 f24 f25 numatomserror gaussnumatoms chi3D
sizeparameter avgtemp f31]=textread(gaussianfitfile,...
```

```
        '%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f','commentstyle','matlab');
        HalfBEC=round(((sigmax+sigmay)./2).*sizefactor./pixelsize);
        xoffsetH=round(positionx./pixelsize);
        yoffsetH=round(positiony./pixelsize);

       xaxisfile=char(strcat(deblank(directory),deblank(basename),deblank('.batch')))
        [e1 xaxis e3 e4 ODTvoltage e6 e7 e8 e9 filedelaytime e11 e12]=textread(xaxisfile,...
        '%s%f%f%s%f%f%f%f%f%f%f%s','commentstyle','matlab');
    end

    if dfgfit==1
        clear fugacityguess
        [fugacityguess]=textread(char(strcat(deblank(batchdirectory),deblank(basename),deblank('fugacityguess.dat'))),'%f','commentstyle','matlab');
        %fugacityguess(2)=5;
        %[fugacityguess]=[1 1 1 1 1 1];
    end


elseif singlefit==1
    file=char(strcat(deblank(directory),deblank(singlebasename),deblank('.dat')));
    dataatom=char('atoms.bny');
    databack=char('back.bny');
end

%OUTPUT FILE
if becfit==0 && dfgfit==0 && becweight==0
    outputdatafile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('2Dfitparams.txt')));
elseif becweight==1 && becfit==0
    outputdatafile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('Excluded2Dfitparams.txt')));
    guessoutputfile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('reducedtempguess.txt')));
elseif becfit==1
    outputdatafile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('BECfitparams.txt')));
elseif dfgfit==1
    outputdatafile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('DFGfitparams.txt')));
    fugacityoutputfile=char(strcat(deblank(batchdirectory),deblank(basename),deblank('fugacity.txt')));
end
%array of 2D fit parameters
%(freq,voltage,odintegralerror,odintegral,etc...)
%deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),
outputfilesfile=char(strcat(deblank(directory),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('.out')));
%array of filenames from input file; currently disabled: see below
outputfiledatamatrix=(strcat(deblank(directory),deblank(basename),deblank('2Dfitparams.xls')));
%array of data similar to outputdatafile (opticaldepths,freqs,odsdev,chi3D,filenumbers,residue1D,#atoms)
%,deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2))
fitdata1D=(strcat(deblank(directory),deblank(basename),deblank('1Dfitparams.txt')));
%array of parameters from fitting routine(delay,density,Nions,amplitude,amplitudedev,centerfreq,centerfreq(dev),linewidth,linewidth(dev),bg,bgdev,bgslope,bgslopedev)
%,deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2))
basenamespectrumplot=(strcat(deblank(directory),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('spec.jpg')));
%jpg file of spectrum and fit
basenamespectrumstatsplot=(strcat(deblank(directory),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('specstats.jpg')));
%jpg file of residuals, chisquared, etc...
if save2places==1
    if becfit==0 && dfgfit==0
        outputdatafile2=char(strcat(deblank(directory2),deblank(basename),deblank('2Dfitparams.txt')));
    elseif becfit==1
        outputdatafile2=char(strcat(deblank(directory2),deblank(basename),deblank('BECfitparams.txt')));
    elseif dfgfit==1
        outputdatafile2=char(strcat(deblank(directory2),deblank(basename),deblank('DFGfitparams.txt')));
    end
    outputfilesfile2=char(strcat(deblank(directory2),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('.out')));
    outputfiledatamatrix2=(strcat(deblank(directory2),deblank(basename),deblank('2Dfitparams.xls')));
    fitdata1D2=(strcat(deblank(directory2),deblank(basename),deblank('1Dfitparams.txt')));
    basenamespectrumplot2=(strcat(deblank(directory2),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('spec.jpg')));
    basenamespectrumstatsplot2=(strcat(deblank(directory2),deblank(basename),deblank('-'),deblank(num2str(r1)),deblank('-'),deblank(num2str(r2)),deblank('specstats.jpg')));
end
if binaryread==1
    dataatom=char('atoms.bny');
    databack=char('back.bny');
elseif binaryread==0
    dataatom=char('atoms.dat');
    databack=char('back.dat');
end

clear opticaldepthsfrom2Dfits
clear alldata
clear alldatares
clear BECfraction
clear ODTPower
clear fbar
clear avgtemp
clear wbar
clear Tc
clear odsdev
clear sdev
clear chi3D
clear sigx3D
clear sigy3D
clear residue1D
clear filenumbers
clear gradmatrix
clear nogradoddata
clear nogradodmatrix
clear errorod
clear errorodmatrix
sumsigx=0;
sumsigy=0;
avgsigx=0;
avgsigy=0;
avgsigz=0;
Nions=0;
% the columns of this matrix are the OD, OD error, chi square 3D, 1D
```

```
% residue no errors

%this loop processes all the data files in this batch
counter=0;
for k=1:q
    s=char(strcat(deblank(directory),deblank(file(k)),deblank(dataatom(1,:))));
    t=char(strcat(deblank(directory),deblank(file(k)),deblank(databack(1,:))));
    counter=counter+1

    if binaryread==1;
        %Read in binary file created in LabView
       fid=fopen(s,'rb','ieee-be'); %Open atoms image
        %LabView saves binary in "big endian" ('be') format: most significant bit in
        %lowest memory address. Matlab needs this info to import binary file correctly.
        fullrawimageatom=fread(fid,matrixsize,'*int16');
        fclose(fid);
        %LabView and Matlab treat matrix coordinates differently.
        %(0,0) for LabView is lower left corner; for Matlab is upper left corner.
        %Therefore, flip about horizontal axis.
        %fullrawimageatom=flipud(fullrawimageatom); %I think this makes it
        %opposite LabView. 2007/01/15

       fid=fopen(t,'rb','ieee-be'); %Open background image
        fullrawimageback=fread(fid,matrixsize,'*int16');
        fclose(fid);
        %fullrawimageback=flipud(fullrawimageback); %I think this makes
       %it
        %opposite LabView. 2007/01/15

    elseif binaryread==0;
        fullrawimageatominter=dlmread(s); %read in data file for atoms
        fullrawimagebackinter=dlmread(t); %read in data file for back
        fullrawimageatom=fullrawimageatominter';
        fullrawimageback=fullrawimagebackinter';
    end

    bgsize=size(fullrawimageback);
    fullsizefull=size(fullrawimageatom);    %Size of atoms matrix
    fullcolumnfull=fullsizefull(1);
    fullrowfull=fullsizefull(2); %no. of rows
    %%%%Region of interest; done before any other array modification.
    %%%%Define in terms of original coordinates (full size array = 1280x1024)
    roiimageatom=fullrawimageatom(roirowstart(k):roirowstop(k),roicolstart(k):roicolstop(k));
    roiimageback=fullrawimageback(roirowstart(k):roirowstop(k),roicolstart(k):roicolstop(k));
    roisizes=size(roiimageatom);
    roicol=roisizes(1);
    roirow=roisizes(2);

    %%%%true software binning
    if softwarebinsize > 1
        for i=1:roicol/softwarebinsize
            for j=1:roirow/softwarebinsize
                rawimageatom(i,j)=sum(sum(roiimageatom(i*softwarebinsize-(softwarebinsize-1):i*softwarebinsize,j*softwarebinsize-(softwarebinsize-1):j*softwarebinsize)));
                rawimageback(i,j)=sum(sum(roiimageback(i*softwarebinsize-(softwarebinsize-1):i*softwarebinsize,j*softwarebinsize-(softwarebinsize-1):j*softwarebinsize)));
            end
        end
    else
        rawimageatom = roiimageatom;
        rawimageback = roiimageback;
    end

    sizefull=size(rawimageatom);    %Size of atoms matrix post-binning
    columnfull=sizefull(1);
    rowfull=sizefull(2);

    if scalebaselevel==1
        sumylowxlow=sum(sum(rawimageatom(1:numberforscale,1:numberforscale)));
        sumylowxhigh=sum(sum(rawimageatom(1:numberforscale,rowfull-numberforscale:rowfull)));
        sumyhighxlow=sum(sum(rawimageatom(columnfull-numberforscale:columnfull,1:numberforscale)));
        sumyhighxhigh=sum(sum(rawimageatom(columnfull-numberforscale:columnfull,rowfull-numberforscale:rowfull)));
        baselevelatom=sumylowxlow+sumylowxhigh+sumyhighxlow+sumyhighxhigh;
        averageylowxlow=sum(sum(rawimageback(1:numberforscale,1:numberforscale)));
        sumylowxhigh=sum(sum(rawimageback(1:numberforscale,rowfull-numberforscale:rowfull)));
        sumyhighxlow=sum(sum(rawimageback(columnfull-numberforscale:columnfull,1:numberforscale)));
        sumyhighxhigh=sum(sum(rawimageback(columnfull-numberforscale:columnfull,rowfull-numberforscale:rowfull)));
        baselevelback=sumylowxlow+sumylowxhigh+sumyhighxlow+sumyhighxhigh;
        rawimageatom=rawimageatom*baselevelback/baselevelatom;
    end %of scalebaselevel

    %%%%Form matrix from only part of the data
    %%%%Sample interleave section; removes every sampleinterleave row or column
    for i=1:columnfull/sampleinterleave
        for j=1:rowfull/sampleinterleave
            for v=1:sampleinterleave
                tempatom(v)=sum(rawimageatom(((i-1)*sampleinterleave+1):i*sampleinterleave,((j-1)*sampleinterleave+v)));
                tempback(v)=sum(rawimageback(((i-1)*sampleinterleave+1):i*sampleinterleave,((j-1)*sampleinterleave+v)));
            end
            rawimageatom2(i,j)=sum(tempatom);
            rawimageback2(i,j)=sum(tempback);
        end
    end
    rawimageatom=rawimageatom2; clear rawimageatom2 %cleaning up array
    rawimageback=rawimageback2; clear rawimageback2 %cleaning up array

    sizefull=size(rawimageback);
    columnfull=sizefull(1); %no. of cols after sampleinterleave
    rowfull=sizefull(2); %no. of rows after sampleinterleave
    for i=1+cutborders:columnfull-cutborders,
        for j=1+cutborders:rowfull-cutborders,
            cutimageback(i-cutborders,j-cutborders)=rawimageback(i,j);
        end
```

```
end

 sizefull=size(rawimageatom);
 columnfull=sizefull(1);
 rowfull=sizefull(2); %no. of rows

 %%%%Cut edges of data by cutborders
 for i=1+cutborders:columnfull-cutborders,
    for j=1+cutborders:rowfull-cutborders,
        cutimageatom(i-cutborders,j-cutborders)=rawimageatom(i,j);
    end
 end
 sizes=size(cutimageatom);
 column=sizes(1);
 row=sizes(2); %no. of rows
 S=zeros(column,row,q);

 %cutimageback=cutimageback(1:column,1:row);
 %sigma=sqrt(cutimageback(:,:));%expected noise, the factor of two acounts for the shot noise of the atom signal and background
 %sigma2=cutimageback(:,:);   ck
 cutimageatom=abs(cutimageatom);
 cutimageback=abs(cutimageback);
 opticaldepthimagesingle=(log(cutimageatom)-log(cutimageback))*(-1);%makes absorption positive
 opticaldepthimagebatch(1:column,1:row,k)=opticaldepthimagesingle; %array of log(data) -log(back)

 %Fit Data
 %Generate guesses and bounds
 %smoothed optical density for finding guesses
 Zguess=filter2(ones(guesssmoothsize,guesssmoothsize),opticaldepthimagesingle(:,:))/guesssmoothsize^2; %atom-back, smoothed lots for finding guesses
 amplitude=max(max(Zguess)); %%this will give trouble if slope is too big
 %guesses for x and y position of cloud; bin by 4 and choose max
 %bin; multiply by 4 to get coordinates in full size array
 if softwarebinsize > 5
    centerfindbinsize=1; %bin by 1 is sufficient in this case because binning was already pretty high.
 elseif softwarebinsize <= 5
    centerfindbinsize=4; %bin by at least 4 because binning was low to begin with
 end
 for i=1:size(Zguess,1)/centerfindbinsize
    for j=1:size(Zguess,2)/centerfindbinsize
        Zguessbinned(i,j)=sum(sum(Zguess(i*centerfindbinsize-(centerfindbinsize-1):i*centerfindbinsize,j*centerfindbinsize-(centerfindbinsize-1):j*centerfindbinsize)));
    end
 end
 [C,I] = max(Zguessbinned); [C2,I2]=max(max(Zguessbinned));
 xoffset=I2*centerfindbinsize;
 yoffset=I(I2)*centerfindbinsize;
 sigx=row/10;
 sigy=column/10;%%%%ONE OF THESE SHOULD BE COLUMN
 averageylowxlow=sum(sum(Zguess(1:5,1:5)))/25;
 averageylowxhigh=sum(sum(Zguess(1:5,row-5:row)))/25;
 averageyhighxlow=sum(sum(Zguess(column-5:column,1:5)))/25;
 averageyhighxhigh=sum(sum(Zguess(column-5:column,row-5:row)))/25;

 offset=(averageylowxlow+averageylowxhigh+averageyhighxlow+averageyhighxhigh)/4 ;
 slopex=((averageylowxhigh+averageyhighxhigh)-(averageylowxlow+averageyhighxlow))/(2*row);
 slopey=((averageyhighxlow+averageyhighxhigh)-(averageylowxlow+averageylowxhigh))/(2*column);

 if becfit ==0 && dfgfit==0
 InitialConditions=[amplitude, sigx, sigy, xoffset, yoffset,offset,slopex,slopey];%
 elseif becfit ==1
 InitialConditions=[amplitude, sigx, sigy, xoffset, yoffset,offset,slopex,slopey];%
 elseif dfgfit ==1
 %fugacityguess(4) = exp(5);
 %fugacityguess(6) = exp(6);

 %fugacityguess(1) = exp(7);
 %fugacityguess(3) = exp(5);
 %fugacityguess(7) = exp(5);
 %fugacityguess(k)=log(fugacityguess(k));
 %fugacityguess(k)=7;
 %[fugacityguess]=[log(10)]
 InitialConditions=[amplitude, sigx, sigy, xoffset, yoffset,offset,slopex,slopey,log(fugacityguess(k))];%
 end
 %pause
 lowerbound=[0, 1, 1, 10, 10, -3, 0, 0,-.01,-.01];%
 upperbound=[3, 50, 50, 90, 90, 3, 1, 1,.01,.01];%
 temp=size(InitialConditions);
 numberofparameters=temp(2);%number of 2d fit parameters

 %Statistics of Image
 %Weird things happen when you calculate these with smoothed images!!!
 %Get estimate of noise, you need to subtract image from background because of fringes and variations in intensity
 average1=sum(sum(cutimageatom(1:limit,1:limit)-cutimageback(1:limit,1:limit)))/limit^2;
 rms1=sqrt(sum(sum((cutimageatom(1:limit,1:limit)-cutimageback(1:limit,1:limit)).^2))/limit^2);
 rmsdeviation1=sqrt(rms1^2-average1^2);

 average2=sum(sum(cutimageatom(1:limit,row-limit:row)-cutimageback(1:limit,row-limit:row)))/limit^2;
 rms2=sqrt(sum(sum((cutimageatom(1:limit,row-limit:row)-cutimageback(1:limit,row-limit:row)).^2))/limit^2);
 rmsdeviation2=sqrt(rms2^2-average2^2);

 average3=sum(sum(cutimageatom(column-limit:column,1:limit)-cutimageback(column-limit:column,1:limit)))/limit^2;
 rms3=sqrt(sum(sum((cutimageatom(column-limit:column,1:limit)-cutimageback(column-limit:column,1:limit)).^2))/limit^2);
 rmsdeviation3=sqrt(rms3^2-average3^2);

 average4=sum(sum(cutimageatom(column-limit:column,row-limit:row)-cutimageback(column-limit:column,row-limit:row)))/limit^2;
 rms4=sqrt(sum(sum((cutimageatom(column-limit:column,row-limit:row)-cutimageback(column-limit:column,row-limit:row)).^2))/limit^2);
 rmsdeviation4=sqrt(rms4^2-average4^2);

 error=abs((rmsdeviation1+rmsdeviation2+rmsdeviation3+rmsdeviation4)/(4))%
 %FIX IMAGINARY ERRORS....THIS IS A PROBLEM!!!!!

 %Create fitting matrices
```

```
 %(sum(sum(cutimageback))/(row*column)) is average value of pixel in cutimageback
 errorodmatrix=0;
 errorodmatrix=ones(row,column).*(error/(sum(sum(cutimageback))/(row*column))); %normalized error - relative to average pixel value
 errorod(1:row*column)=error/(sum(sum(cutimageback))/(row*column)); %normalized error - relative to average pixel value
 %%----------------------------------------------------------------------%%
 %Change weight of BEC portion of cloud
if becweight==1
 if fit==1
     for u=xoffsetH(k)-HalfBEC(k):1:xoffsetH(k)+HalfBEC(k)
         for v=yoffsetH(k)-HalfBEC(k):1:yoffsetH(k)+HalfBEC(k)
            errorodmatrix(u,v)=BECweight;
         end
     end
 end
end
  %%----------------------------------------------------------------------%%
 errorod(1:row*column)=errorodmatrix(:);
 xvec=1:row;
 yvec=1:column;%%%%ONE OF THESE SHOULD BE COLUMN
 [xdata,ydata]=meshgrid(xvec,yvec);

 numberofpoints=column*row*ones(column,row);%matrix in which each entry is the number of points
 coordmatrix=[xdata(:),ydata(:),errorod(:),numberofpoints(:)];

if fit==1 && column*row<100000; % skip fit if fit==0 or matrix is larger than 100,000 pixels (very slow otherwise)
    %actual fit
    Z2=(opticaldepthimagesingle(:)./(errorod(:)*sqrt(column*row)));  %flattened(1D) list of image weighted by uncertainties, not smoothed
    if becfit==0 && dfgfit==0
    [P,r,J]=nlinfit(coordmatrix,Z2(:),@gaussfitfunctionerrorsslope,InitialConditions); %fitting statement - expecting data divided by weight and sqrt(number of points)
    elseif becfit==1
     [P,r,J]=nlinfit(coordmatridropx,Z2(:),@BECfitfunctionerrorsslope,InitialConditions); %fitting statement - expecting data divided by weight and sqrt(number of points)
    elseif dfgfit==1
     [P,r,J]=nlinfit(coordmatrix,Z2(:),@FermiFitFunction,InitialConditions); %fitting statement - expecting data divided by weight and sqrt(number of points)
    end

    %%% Extract the temperature from thermal pedestal
    xsizetemp(k) = ((P(2).*sizefactor).^2.*mass)./(kBoltz.*(droptime./timefactor).^2)./templow; %calculate temperature based on x size
    ysizetemp(k) = ((P(3).*sizefactor).^2.*mass)./(kBoltz.*(droptime./timefactor).^2)./templow; %calculate temperature based on y size
    avgtemp(k) = (xsizetemp(k)+ysizetemp(k))./2; %1000 to get in nanokelvin


    if dfgfit==0
        fugacity(k)=0; %dummy since we dont use fugacity here
    elseif dfgfit==1
        fugacity(k)=exp(P(9));
    end

    ci=nlparci(P,r,J); %error in the fit parameters 95% confidence interval
       %%saveci=zeros(1,6,2);
    saveci(k,:,:)=ci;
    sdev=(saveci(:,:,2)-saveci(:,:,1))/4;%turn confidence interval into 1 sigma
    %%%%%%%THIS MUST BE RECORDED SOMEWHERE< ALONG WITH ALL FIT PARAMETERS
    %odsdev=sdev(:,1);
    %second derivative of chisquared wrt amplitude at fit paramters
    %open 'Dgaussfitfunctionrf.m'%derivative of chi squared, not yet summed over all points
    %presumdervchi3D=Dgaussfitfunctionrf(P, coordmatrix); % this is the second dervative of chi3d with respect to amplitude, not yet summed over all points
    %dervchi3(counter)=sum(presumdervchi3D);
    %odsdev2(counter)= sqrt(2)*sqrt(1/((column*row)*dervchi3(counter)));
    %error=.01
    %temp=size(smoothopticaldepthimagesingle)
    %Z3 = smoothopticaldepthimagesingle(:)./errorod/8100

    %Below are the calculations of residues and chi sqares considering and not considering errors for the 3D fits
    if becfit==0 && dfgfit==0 && becweight==0
        clear numberconstraints
        numberconstraints = 8;
        weightedresidue3D=(Z2(:)-gaussfitfunctionerrorsslope(P, coordmatrix)); %weighted residues
        weightedresidue3D=weightedresidue3D*sqrt(numberofpoints(1:1)); % fix so real reduced chi-square can be computed
    elseif becfit==0 && becweight==1
        clear numberconstraints
        numberconstraints = 8;
        weightedresidue3D=(Z2(:)-gaussfitfunctionerrorsslope(P, coordmatrix));
        weightedresidue3D=weightedresidue3D*sqrt(numberofpoints(1:1)); % fix so real reduced chi-square can be computed
    elseif becfit==1 && becweight==1
        weightedresidue3D=(Z2(:).*errorod(:)-BECfitfunctionerrorsslope(P, coordmatrix).*errorod(:)); %weighted residues
    elseif dfgfit==1
        clear numberconstraints
        numberconstraints=9;
        weightedresidue3D=(Z2(:)-FermiFitFunction(P, coordmatrix)); %weighted residues
        weightedresidue3D=weightedresidue3D*sqrt(numberofpoints(1:1)); % fix so real reduced chi-square can be computed
    end
    %chi3D(k)=(transpose(weightedresidue3D)*weightedresidue3D);
    chi3D(k)=(transpose(weightedresidue3D)*weightedresidue3D)/(numberofpoints(1:1)-numberconstraints)
    %weightedresidue3Dnoerrors=(opticaldepthimagesingle(:)-gaussfitfunctionnoerrorsslope(P, coordmatrix));
    %chi3Dnoerrors(counter)=(transpose(weightedresidue3Dnoerrors)*weightedresidue3Dnoerrors)*(1/(column*row))*(1/errorod(1))^2

    %residue=cumsum(((smoothopticaldepthimagesingle(:)-gaussfitfunctionrf(P, coordmatrix))./sigma(:)).^2);
    %sqrt(residue(8100))/(8100)  %chisquared
    %form output parameters
    %fraction absorption is p(1)

    %removing slope and background

    if becfit ==1
       %P(6)=0;
       %P(7)=0;
       %P(8)=0;
    end


    gradmatrix=(xdata(:)-P(4))*P(7)+(ydata(:)-P(5))*P(8)+P(6);
```

```
            nogradoddata3=opticaldepthimagesingle(:)-gradmatrix(:);             %The total number
%             %nogradoddata1=gaussfitfunctionerrorsslope(P, coordmatrix).*(errorod(:)*sqrt(column*row));            %The total number
%             nogradoddata1=gaussfitfunctionerrorsslope(P, coordmatrix);            %The total number
% %             nogradoddata3=(gaussfitfunctionerrorsslope(P, coordmatrix));

          %Count on the number below fitting curves
          if becfit==0 && dfgfit == 0
              nogradoddata=(gaussfitfunctionerrorsslope(P, coordmatrix).*(errorod(:)*sqrt(column*row))-gradmatrix(:));
          elseif becfit==1
              nogradoddata=(BECfitfunctionerrorsslope(P, coordmatrix).*(errorod(:)*sqrt(column*row))-gradmatrix(:));
          elseif dfgfit ==1
              nogradoddata=(FermiFitFunction(P, coordmatrix).*(errorod(:)*sqrt(column*row))-gradmatrix(:));
          end
          aaa= errorod(:);
          nogradodmatrix = reshape(nogradoddata, size(xdata));
          nogradoddata1=nogradoddata;

          pixelcount=0;
          odsum=0;
          odaverage=0;
          totalod=0;
          odintegral=0;
          %Sum the optical depth to get total optical depth
          for i=1:row
              for j=1:column
                  tempindex=(i-1)*column+j;
                  totalod=totalod+nogradoddata(tempindex);
              end
          end
          odintegralerror=errorod(1)*sqrt(row*column)*(prebinning*pixelsize)^2;
          %pixelcount
          %odaverage=odsum/pixelcount
          odintegral=totalod*(prebinning*pixelsize)^2;

          %odsdev(k,1)=error/(sum(sum(cutimageback))/(row*column))/sqrt(pixelcount);
          odsdev(k,1)=odintegralerror;

          P; %fit parameters
          onesigma = (ci(:,2)-ci(:,1))./4; %1-sigma "error" for fit parameters
          output(1:1,k,basenamenumber)=totalod;%total od
          output(2:2,k,basenamenumber)=P(1);%od average for selected region (peak od?)
          output(3:3,k,basenamenumber)=pixelsize*prebinning*P(4);%positionx in meters
          output(4:4,k,basenamenumber)=pixelsize*prebinning*P(5);%positiony in meters
          sigx3D(k)=pixelsize*prebinning*abs(P(2));%sigmax in meters
          sigy3D(k)=pixelsize*prebinning*abs(P(3));%sigmay in meters
          sigxerror3D(k)=pixelsize*prebinning*abs(sdev(k,2));%sigmaxerror in meters
          sigyerror3D(k)=pixelsize*prebinning*abs(sdev(k,3));%sigmayerror in meters
          sumsigx=sumsigx+pixelsize*prebinning*abs(P(2));
          sumsigy=sumsigy+pixelsize*prebinning*abs(P(3));
          output(5:5,k,basenamenumber)= odintegral;%absorption fraction, or more accurately - optical depth

          for j=1:numberofparameters
              output(5+j,k,basenamenumber)=P(j);%write fit information into output
          end
          for jj=1:numberofparameters
              output(5+numberofparameters+jj,k,basenamenumber)=(ci(jj,2)-ci(jj,1))./4; %95% ci for all fit parameters will go into output file
          end
%             if becfit==1
%               for jj=numberofparameters:numberofparameters
%                 output(5+numberofparameters+jj,k,basenamenumber)=0; %95% ci for all fit parameters will go into output file
%               end
%             end
          output(1:1,k,basenamenumber);
          output(2:2,k,basenamenumber);
          output(3:3,k,basenamenumber);
          output(4:4,k,basenamenumber);
          output(5:5,k,basenamenumber);
          %output(6:(6+2*numberofparameters),k,basenamenumber);
          %             for j=1:numberofparameters
          %                 lowererror(j,k,basenamenumber)=ci(j);
          %             end
          %              for j=numberofparameters+1:2*numberofparameters
          %                 uppererror(j,k,basenamenumber)=ci(j); %this seems wrong
          %             end
          %for j=1:5

          %    output(13+j,k,basenamenumber)=P(j);%write fit information into output

          %      lowererror(j,k,basenamenumber)=ci(j);
          %  end
      elseif column*row >= 100000
          error('analysis stopped because array is larger than 100,000 pixels')
      end% ends the if fit==0/1 , skip fit if fit==0
      %%%%%%%%%WHY NO TEXT FOR K>6 It seems if it
      %has a figure number over 7 => no text!????????
      b=ones(smoothplotsize,smoothplotsize);
      smoothopticaldepthimagesingle=filter2(b,opticaldepthimagesingle(:,:))/smoothplotsize^2; %atom-back, smoothed a bit for fitting
      if plotevolution==1
        figure(2222)
        subplot(5,5,k)
        surf(smoothopticaldepthimagesingle)
        hold on
        xlim([0 row])
        ylim([0 column])
        set(gca,'View',[0 90])
        shading flat
        grid off
        title(strcat(num2str(imagevco(k)),'ms'));
        %hold off
      end
      if plotfits==1;
```

```
%%%%%smooth data forplotting
b=ones(smoothplotsize,smoothplotsize);
%opticaldepthimagesingle is optical depth log(cutimageatom/cutimageback)
smoothopticaldepthimagesingle=filter2(b,opticaldepthimagesingle(:,:))/smoothplotsize^2; %atom-back, smoothed a bit for fitting

if standalone==1
  figure(400+k) %Plot raw data (atoms - back)
   surf(smoothopticaldepthimagesingle)
  hold on
  set(gca,'View',[0 90])
  shading flat
  grid off
  axis off
  xlabel('x');
  ylabel('y');
  title(file(k));
  hold off
  %            elseif standalone==2
  %              figure(400+k) % Create a new figure ...
  %                %plot smoothed data, even though we fit unsmoothed
  %                legend('name(k,8:namelength)')
  %                subplot(2, 2, 1) % plotdata
  %                 surf(smoothopticaldepthimagesingle, 'Marker', '.'); % Plot data
  %               colorbar
  %               shading flat
  %               axis off;
  %               grid off;
  %               xlabel('x')
  %               ylabel('y')
  %               title(file(k))
   %                legend('',num2str(batchvcovoltage(k,basenamenumber)),0)
end %end of standalone case statement


if fit==1  % skip plot of fit if fit==0

  figure(500+k) %Plot raw data (atoms - back)
   subplot(2,2,1)
   surf(smoothopticaldepthimagesingle)
  hold on
  set(gca,'View',[0 90])
  shading flat
  grid off
  axis off
   title(basename,'FontSize',axisfontsize)
  xlabel('x');
  ylabel('y');
  xlim([0 row]);
  ylim([0 column]);
  % title(file(k));
  hold off

   %%%%%%%%%PLACEMENT OF CONSTANTS FOR RESIDUAL PLOTS
   xresidualconstants=0;
   yresidualconstants=column*pixelconv*2;
   zresidualconstants=1.5*(P(1));
   %%%%%%%%%%%%%%
   subplot(2, 2, 2) % plot fit

   %%%Generate plot of optical density based on fitted parameters, remove distortion of data due to dividing by weight and sqrt(number of points)
   errorod(1:row*column)=1;
   numberofpoints=ones(column,row);%
   coordmatrix=[xdata(:),ydata(:),errorod(:),numberofpoints(:)];
  if becfit==1
   opticaldepthcalcvector=BECfitfunctionerrorsslope(P, coordmatrix);%optical depth calculated from fit parameters (NOT WEIGHTED)
  elseif becfit==0 && dfgfit==0
      opticaldepthcalcvector=gaussfitfunctionerrorsslope(P, coordmatrix);
  elseif dfgfit==1
      opticaldepthcalcvector=FermiFitFunction(P, coordmatrix);
  end
   opticaldepthcalcmatrix = reshape(opticaldepthcalcvector, size(xdata));

   surf(pixelconv*xvec,pixelconv*yvec,opticaldepthcalcmatrix, 'Marker', '.'); % Plot fit
   zlabel('Optical Depth','FontSize',axisfontsize);
   xlabel('X (microns)','FontSize',axisfontsize);
   ylabel('Y (microns)','FontSize',axisfontsize);
   set(gca,'FontSize', axisfontsize);
   text(xresidualconstants,yresidualconstants,zresidualconstants, strcat(char('Peak OD ='), num2str(P(1)), ' \pm ', num2str(sdev(k,1))),'FontSize',textsize);
   text(xresidualconstants,yresidualconstants,zresidualconstants -P(1)/4, strcat(char('\sigma_x ='), num2str(sigx3D(k)), ' \pm ', num2str(sigxerror3D(k))),'FontSize',textsize);
   text(xresidualconstants,yresidualconstants,zresidualconstants -P(1)/2, strcat(char('\sigma_y ='), num2str(sigy3D(k)), ' \pm ', num2str(sigyerror3D(k))),'FontSize',textsize);
   colorbar
   shading flat
   %axis off;
   grid off;
   subplot(2, 2, 3) %plot residuals
   %form smoothed residuals
   residuals=smoothopticaldepthimagesingle-opticaldepthcalcmatrix;
   temp=reshape(residuals,size(xdata)); %put in matrix form
   smoothresiduals=filter2(b,temp(:,:))/smoothplotsize^2; %residue smoothed a bit for fitting
   surf(pixelconv*xvec,pixelconv*yvec,smoothresiduals, 'Marker', '.');
   zlabel('Optical Depth','FontSize',axisfontsize);
   xlabel('X (microns)','FontSize',axisfontsize);
   ylabel('Y (microns)','FontSize',axisfontsize);
   set(gca,'FontSize', axisfontsize);
   view([0 90])
   caxis('auto');
   title('true residues (NOT weighted by uncertainties)','FontSize',axisfontsize)
   colorbar
   shading flat
   %axis off;
   grid off;
```

```
            subplot(2, 2, 4) % plot weighted residuals
            %form smoothed residuals
            temp=reshape(weightedresidue3D,size(xdata)); %put in matrix form
            smoothweightedresidue3D=filter2(b,temp(:,:))/smoothplotsize^2; %weighted residue smoothed a bit for fitting

            surf(pixelconv*xvec,pixelconv*yvec,smoothweightedresidue3D, 'Marker', '.');
            zlabel('Optical Depth','FontSize',axisfontsize);
            xlabel('X (microns)','FontSize',axisfontsize);
            ylabel('Y (microns)','FontSize',axisfontsize);
            set(gca,'FontSize', axisfontsize);
            % view([0 90])
            caxis('auto');
             title('residues weighted by uncertainties','FontSize',axisfontsize)
            colorbar
            shading flat
            %axis off;
            grid off;

         end % end the if for: skip plot of fit if fit==0

      end %end plot if loop

      if plotdensity==1
        figure(2000+filecounter)
         %P(9) = -40;
         xoptimal=floor(P(4));
         yoptimal=floor(P(5));
         if becfit==0 && dfgfit==0
          yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*exp(-(((xoptimal-P(4)).^2)/(2*P(2)^2))-(((yvec-P(5)).^2)/(2*P(3)^2))));%./
(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
          xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*exp(-(((xvec-P(4)).^2)/(2*P(2)^2))-(((yoptimal-P(5)).^2)/(2*P(3)^2))));
         elseif becfit==1
          yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*heaviside(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./P(3)).^2).*(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./P(3)).^2).^(3/2)));
%./(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
          xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*heaviside(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./P(3)).^2).*(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./P(3)).^2).^(3/2)));
         elseif dfgfit==1
          yslice=(P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+(P(1)*dilog(-exp(P(9)).*exp(-((xoptimal-P(4)).^2./(2*P(2)^2))).*exp(-((yvec-P(5)).^2/(2*P(3).^2)))))/dilog(-exp(P(9))))
          xslice=(P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+(P(1)*dilog(-exp(P(9)).*exp(-((xvec-P(4)).^2./(2.*P(2).^2))).*exp(-((yoptimal-P(5)).^2/(2.*P(3).^2)))))/dilog(-exp(P(9))))

         end
         subplot(5,5,k)
         plot((Zguess(floor(P(5))-1,:)+Zguess(floor(P(5)),:)+Zguess(floor(P(5))+1,:))/3,'.b')
         hold on
         plot(xvec,xslice,'-b')
         hold on
         plot(xvec,errorodmatrix(:,yoffset)./BECweight,'-b')
         hold on
         plot((Zguess(:,floor(P(4))-1)+Zguess(:,floor(P(4)))+Zguess(:,floor(P(4))+1))/3+1,'.r')
         hold on
         plot(yvec,errorodmatrix(xoffset,:)./BECweight+1,'-r')
         hold on
         plot(yvec,yslice+1,'-r')
         hold on
         xlim([0 row])
         %xlim([0 40])
         ylim([0 2])
         xlabel('x and y position');
         ylabel('Normalized Density');
         if k==8
            legend('x','y','Position','EastOutside');
         end
         title(strcat(num2str(imagevco(k)),'ms'));
         hold off
          %Z =((offset+slopex*(x-xoffset)+slopey*(y-yoffset)+amplitude*exp(-(((x- xoffset).^2)/(2*sigx^2))-(((y- yoffset).^2)/(2*sigy^2))))./(w*sqrtnumbpoints)); %

         figure(3000+filecounter)
         xoptimal=floor(P(4));
         yoptimal=floor(P(5));
         if becfit==0
          yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*exp(-(((xoptimal-P(4)).^2)/(2*P(2)^2))-(((yvec-P(5)).^2)/(2*P(3)^2))));%./
(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
          xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*exp(-(((xvec-P(4)).^2)/(2*P(2)^2))-(((yoptimal-P(5)).^2)/(2*P(3)^2))));
         elseif becfit==1
          yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*heaviside(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./P(3)).^2).*(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./P(3)).^2).^(3/2)));
%./(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
          xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*heaviside(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./P(3)).^2).*(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./P(3)).^2).^(3/2)));
         end
         subplot(5,5,k)
         % plot((Zguess(floor(P(5))-1,:)+Zguess(floor(P(5)),:)+Zguess(floor(P(5))+1,:))/3-xslice,'.b')
         plot(Zguess(floor(P(5)),:)-xslice,'.b')

         hold on
         %plot(xvec,xslice,'-b')
         hold on
         plot(xvec,errorodmatrix(:,yoffset)./BECweight/10,'-b')
         hold on
         %plot((Zguess(:,floor(P(4))-1)+Zguess(:,floor(P(4)))+Zguess(:,floor(P(4))+1))/3-yslice(:),'.r')
         plot(Zguess(:,floor(P(4)))-yslice(:)+.5,'.r')

         hold on
         plot(yvec,errorodmatrix(xoffset,:)./BECweight/10+.5,'-r')
         hold on
         %plot(yvec,yslice,'-r')
         hold on
         xlim([0 row])
         %xlim([20 60])
         ylim([0 2])
         xlabel('x and y position');
         ylabel('Normalized Density');
         if k==8
```

```
        legend('x','y','Position','EastOutside');
    end
    title(strcat(num2str(imagevco(k)),'ms'));
    hold off
end %end of plotdensity case structure

%%%%Collect various data and write to outputdatafile, but only if fit is on.
if fit==1 && column*row<100000; % skip fit if fit==0
    %add data for this file to the alldata matrix
    %detuning = dimage(basenamenumber)*10^6;      % image beam detuning in Hz
    absorptioncross = 6*pi*(lambda/(2*pi))^2/(1+(2*detuning/naturalwidth)^2);     % absorption cross section; used to calculate #atoms
  if becfit==0 && dfgfit==0
    numatoms = odintegral./absorptioncross;     % number of atoms obtained from the optical depth

  elseif becfit==1
        numatoms=P(1)*abs(P(2))*abs(P(3))*pixelsize*pixelsize*(3/4)*(8*pi/15)/(absorptioncross);

  elseif dfgfit==1
      numatoms=0; %Not determining number of atoms from fermi-dirac fits yet

  end
    numatomserror = odintegralerror.*numatoms./odintegral;  % dN = N*dOD/OD
    %numatomserror = odintegralerror./absorptioncross;      % dN = dOD/absorptioncross

    %frequencies not important to non-PAS experiments now.
    %dshift = aomvector(basenamenumber)*1000;
    dcat(k)=(-.02288*(vcovoltage(k)^4)+0.497*(vcovoltage(k)^3)-3.436*(vcovoltage(k)^2)+23.93*vcovoltage(k)+169);
    %alldata(k,1)=(dshift(basenamenumber)-dsatabs(basenamenumber)+2*(dzeeman(basenamenumber)+dPAS+(catseyesign(basenamenumber)*dcat(k))))
    alldata(k,1)=(dshift+2*(dzeeman+2*dPAS-dsatabs+(catseyesign*dcat(k))));
    alldata(k,2)=batchvcovoltage(k,basenamenumber);
    alldata(k,3)=odintegralerror;
    alldata(k,4)=odintegral;
    for index=1:(size(output,1));
        alldata(k,index+4)=output(index,k,basenamenumber);
    end; %fit parameters and 1-sigma error for each parameter
    alldata(k,4+size(output,1)+1)=numatomserror;
    alldata(k,4+size(output,1)+2)=numatoms; %Number of atoms is useful
    alldata(k,4+size(output,1)+3)=chi3D(k);
    alldata(k,4+size(output,1)+4)=pixelsize*prebinning; %size parameter saved so hardware and software binning settings only set once in imagefit.
    alldata(k,4+size(output,1)+5)=avgtemp(k);
    alldata(k,4+size(output,1)+6)=fugacity(k);
    temp=size(char(file(k,:)));
    allfiles(k)=file(k);

    %write all 2D fit parameters to outputdatafile
    dlmwrite(outputdatafile, alldata,'\t');

    if fit==1 && becweight==1
        NumberCorrection=2;
        PowerFactor=0.91;
        ODTPower=ODTVoltage*PowerFactor;
        fbarModel100 = -0.002*ODTPower.^6 + 0.074*ODTPower.^5 - 1.049*ODTPower.^4 + 7.792*ODTPower.^3 - 33.39*ODTPower.^2 + 100.1*ODTPower - 2.304;
        Tfermi=((6.*gaussnumatoms.*NumberCorrection./10).^(1/3).*hbar.*(2.*pi.*fbarModel100))./kBoltz; %fermi temperature calculated from model of trap
        guessdata(k)=avgtemp(k)*10^-6/Tfermi(k);

        dlmwrite(guessoutputfile,guessdata,'\t');

    end


    if dfgfit==1
      fugacitydata(k,1)=imagevco(k);
      fugacitydata(k,2)=fugacity(k);
      fugacitydata(k,3)=exp(saveci(k,9,1));
      fugacitydata(k,4)=exp(saveci(k,9,2));
      fugacitydata(k,5)=exp(saveci(k,9,1));
      fugacitydata(k,6)=exp(saveci(k,9,2));

      dlmwrite(fugacityoutputfile,fugacitydata,'\t');
    end

    if save2places==1
        dlmwrite(outputdatafile2, alldata,'\t');
    end
    %Columns for alldata
    %1     frequencies (MHz)
    %2     batchvcovoltage
    %3     odintegralerror
    %4     odintegral
    %5     totalod
    %6     od average for selected region
    %7     position x in meters
    %8     position y in meters
    %9     odintegral
    %10    raw 2d output P(1) peak OD
    %11    raw 2d output P(2)sig x in pixels
    %12    raw 2d output P(3)sig y in pixels
    %13    raw 2d output P(4)
    %14    raw 2d output P(5)
    %15 raw 2d output P(6)
    %16 raw 2d output P(7)
    %17 raw 2d output P(8)
    %18 raw 2d output 1-sigma for P(1)
    %19 raw 2d output 1-sigma for P(2)
    %20 raw 2d output 1-sigma for P(3)
    %21 raw 2d output 1-sigma for P(4)
    %22 raw 2d output 1-sigma for P(5)
    %23 raw 2d output 1-sigma for P(6)
    %24 raw 2d output 1-sigma for P(7)
    %25 raw 2d output 1-sigma for P(8)
    %26    number of atoms error
```

```
    %27   number of atoms
    %28 chi3D
    %29 size parameter (pixelsize*prebinning)
    %30 temperature extracted from the thermal atoms in pedestal
    %dlmwrite(outputfilesfile, allfiles,'\t')

    if k>q

    else
      figure(1)
       subplot(2,2,1)
        plot(imagevco(1:counter),alldata(:,27)./10^6,'*b')
       hold on
       %xlabel('DAC voltage');
       set(gca, 'FontSize', axisfontsize);
       ylabel('Number of atoms (10^6)');
       title(char(basename));
       subplot(2,2,2)
       hold on
        plot(imagevco(1:counter),alldata(:,10),'*b')
       set(gca, 'FontSize', axisfontsize);
       ylabel('Peak OD');
       subplot(2,2,3)
       hold on
        plot(imagevco(1:counter),sigx3D(1:counter),'*b')
       hold on
        plot(imagevco(1:counter),sigy3D(1:counter),'dr','MarkerFaceColor','r')

       set(gca, 'FontSize', axisfontsize);
       %xlabel('DAC voltage');
       ylabel('RMS width X (blue Y(red)');
       xlabel('DAC voltage');
       %title('frequency calibration');
       subplot(2,2,4)
       hold on
        plot(imagevco(1:counter),avgtemp(1:counter),'*b')
       set(gca, 'FontSize', axisfontsize);
       xlabel('DAC voltage');
       ylabel('Average Temperature');
      hold off
    end
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if fitresiduals==1
         residuals=smoothopticaldepthimagesingle-opticaldepthcalcmatrix;
         temp=reshape(residuals,size(xdata)); %put in matrix form
        smoothresiduals=filter2(b,temp(:,:))/smoothplotsize^2; %residue smoothed a bit for fitting
        residualsforfitting=smoothresiduals;
        Zguess=residualsforfitting;
amplitude=max(max(Zguess)); %%this will give trouble if slope is too big
%guesses for x and y position of cloud; bin by 4 and choose max
%bin; multiply by 4 to get coordinates in full size array
if softwarebinsize > 5
   centerfindbinsize=1; %bin by 1 is sufficient in this case because binning was already pretty high.
elseif softwarebinsize <= 5
   centerfindbinsize=4; %bin by at least 4 because binning was low to begin with
end
for i=1:size(Zguess,1)/centerfindbinsize
    for j=1:size(Zguess,2)/centerfindbinsize
        Zguessbinned(i,j)=sum(sum(Zguess(i*centerfindbinsize-(centerfindbinsize-1):i*centerfindbinsize,j*centerfindbinsize-(centerfindbinsize-1):j*centerfindbinsize)));
   end
end
[C,I] = max(Zguessbinned); [C2,I2]=max(max(Zguessbinned));
xoffset=I2*centerfindbinsize;
yoffset=I(I2)*centerfindbinsize;
sigx=row/10;
sigy=column/10;%%%%ONE OF THESE SHOULD BE COLUMN
averageylowxlow=sum(sum(Zguess(1:5,1:5)))/25;
averageylowxhigh=sum(sum(Zguess(1:5,row-5:row)))/25;
averageyhighxlow=sum(sum(Zguess(column-5:column,1:5)))/25;
averageyhighxhigh=sum(sum(Zguess(column-5:column,row-5:row)))/25;

offset=(averageylowxlow+averageylowxhigh+averageyhighxlow+averageyhighxhigh)/4 ;
slopex=((averageylowxhigh+averageyhighxhigh)-(averageylowxlow+averageyhighxlow))/(2*row);
slopey=((averageyhighxlow+averageyhighxhigh)-(averageylowxlow+averageylowxhigh))/(2*column);

if becfit ==0
InitialConditions=[amplitude, sigx, sigy, xoffset, yoffset,offset,slopex,slopey];%
elseif becfit ==1
InitialConditions=[amplitude, sigx, sigy, xoffset, yoffset,offset,slopex,slopey];%
end
%pause
lowerbound=[0, 1, 1, 10, 10, -3, 0, 0,-.01,-.01];%
upperbound=[3, 50, 50, 90, 90, 3, 1, 1,.01,.01];%
temp=size(InitialConditions);
numberofparameters=temp(2);%number of 2d fit parameters

%Statistics of Image
%Weird things happen when you calculate these with smoothed images!!!
%Get estimate of noise, you need to subtract image from background because of fringes and variations in intensity
average1=sum(sum(cutimageatom(1:limit,1:limit)-cutimageback(1:limit,1:limit)))/limit^2;
rms1=sqrt(sum(sum((cutimageatom(1:limit,1:limit)-cutimageback(1:limit,1:limit)).^2))/limit^2);
rmsdeviation1=sqrt(rms1^2-average1^2);

average2=sum(sum(cutimageatom(1:limit,row-limit:row)-cutimageback(1:limit,row-limit:row)))/limit^2;
rms2=sqrt(sum(sum((cutimageatom(1:limit,row-limit:row)-cutimageback(1:limit,row-limit:row)).^2))/limit^2);
rmsdeviation2=sqrt(rms2^2-average2^2);

average3=sum(sum(cutimageatom(column-limit:column,1:limit)-cutimageback(column-limit:column,1:limit)))/limit^2;
rms3=sqrt(sum(sum((cutimageatom(column-limit:column,1:limit)-cutimageback(column-limit:column,1:limit)).^2))/limit^2);
rmsdeviation3=sqrt(rms3^2-average3^2);
```

```
 average4=sum(sum(cutimageatom(column-limit:column,row-limit:row)-cutimageback(column-limit:column,row-limit:row)))/limit^2;
 rms4=sqrt(sum(sum((cutimageatom(column-limit:column,row-limit:row)-cutimageback(column-limit:column,row-limit:row)).^2))/limit^2);
 rmsdeviation4=sqrt(rms4^2-average4^2);

 error=abs((rmsdeviation1+rmsdeviation2+rmsdeviation3+rmsdeviation4)/(4))%
 %FIX IMAGINARY ERRORS....THIS IS A PROBLEM!!!!!

 %Create fitting matrices
 %(sum(sum(cutimageback))/(row*column)) is average value of pixel in cutimageback
 errorodmatrix=0;
 errorodmatrix=ones(row,column).*(error/(sum(sum(cutimageback))/(row*column))); %normalized error - relative to average pixel value
 errorod(1:row*column)=error/(sum(sum(cutimageback))/(row*column)); %normalized error - relative to average pixel value
 %%--------------------------------------------------------------%%
 %Change weight of BEC portion of cloud
if becweight==0
 if fit==1
     for u=xoffsetH(k)-HalfBEC(k):1:xoffsetH(k)+HalfBEC(k)
         for v=yoffsetH(k)-HalfBEC(k):1:yoffsetH(k)+HalfBEC(k)
             errorodmatrix(u,v)=BECweight;
         end
     end
 end
end
  %%--------------------------------------------------------------%%
 errorod(1:row*column)=errorodmatrix(:);
 xvec=1:row;
 yvec=1:column;%%%%ONE OF THESE SHOULD BE COLUMN
 [xdata,ydata]=meshgrid(xvec,yvec);

 numberofpoints=column*row*ones(column,row);%matrix in which each entry is the number of points
 coordmatrix=[xdata(:),ydata(:),errorod(:),numberofpoints(:)];
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             sizes=size(residualsforfitting);
%             column=sizes(1);
%             row=sizes(2); %no. of rows
%              errorod(1:row*column)=error/(sum(sum(cutimageback))/(row*column)); %normalized error - relative to average pixel value
%
%            xvec=1:row;
%             yvec=1:column;%%%%ONE OF THESE SHOULD BE COLUMN
%             [xdata,ydata]=meshgrid(xvec,yvec);
%
%              numberofpoints=column*row*ones(column,row);%matrix in which each entry is the number of points
%             coordmatrix=[xdata(:),ydata(:),errorod(:),numberofpoints(:)];
%              InitialConditions=[amplitude, sigx/10, sigy/10,P(4),P(5),0,0,0];
%
%            %actual fit
%             Z2=(residualsforfitting(:)./(errorod(:)*sqrt(column*row)));  %flattened(1D) list of image weighted by uncertainties, not smoothed
%
    Z2=(residualsforfitting(:)./(errorod(:)*sqrt(column*row)));  %flattened(1D) list of image weighted by uncertainties, not smoothed
        if becfit==0
    [P,r,J]=nlinfit(coordmatrix,Z2(:),@BECfitfunctionerrorsslope,InitialConditions); %fitting statement - expecting data divided by weight and sqrt(number of points)
       elseif becfit==1
     [P,r,J]=nlinfit(coordmatrix,Z2(:),@gaussfitfunctionerrorsslope,InitialConditions); %fitting statement - expecting data divided by weight and sqrt(number of points)
      end

       ci=nlparci(P,r,J); %error in the fit parameters 95% confidence interval
       %%saveci=zeros(1,6,2);
      savecires(k,:,:)=ci;
       sdev=(savecires(:,:,2)-savecires(:,:,1))/4;%turn confidence interval into 1 sigma
        %Below are the calculations of residues and chi sqares considering and not considering errors for the 3D fits
      if becfit==0
      weightedresidue3D=(Z2(:)-BECfitfunctionerrorsslope(P, coordmatrix)); %weighted residues
       elseif becfit==1
     weightedresidue3D=(Z2(:)-gaussfitfunctionerrorsslope(P, coordmatrix)); %weighted residues
      end

        chi3Dres(k)=(transpose(weightedresidue3D)*weightedresidue3D);
        %removing slope and background
       gradmatrix=(xdata(:)-P(4))*P(7)+(ydata(:)-P(5))*P(8)+P(6);
        %nogradoddata=residualsforfitting(:)-gradmatrix(:);
      if becfit==1
       nogradoddata=(gaussfitfunctionerrorsslope(P, coordmatrix)-gradmatrix(:));
      elseif becfit==0
       nogradoddata=(BECfitfunctionerrorsslope(P, coordmatrix)-gradmatrix(:));
      end
        nogradodmatrix = reshape(nogradoddata, size(xdata));
        nogradoddata2=nogradoddata;


       pixelcount=0;
      odsum=0;
       odaverage=0;
      totalod=0;
       odintegral=0;
       %Sum the optical depth to get total optical depth
      for i=1:row
          for j=1:column
              tempindex=(i-1)*column+j;
               totalod=totalod+nogradoddata(tempindex);
          end
      end
        odintegralerror=errorod(1)*sqrt(row*column)*(prebinning*pixelsize)^2;
       %pixelcount
        %odaverage=odsum/pixelcount
        odintegral=totalod*(prebinning*pixelsize)^2;

        %odsdev(k,1)=error/(sum(sum(cutimageback))/(row*column))/sqrt(pixelcount);
       odsdevres(k,1)=odintegralerror;

      P; %fit parameters
```

```
 onesigma = (ci(:,2)-ci(:,1))./4; %1-sigma "error" for fit parameters
 outputres(1:1,k,basenamenumber)=totalod;%total od
 outputres(2:2,k,basenamenumber)=P(1);%od average for selected region (peak od?)
 outputres(3:3,k,basenamenumber)=pixelsize*prebinning*P(4);%positionx in meters
 outputres(4:4,k,basenamenumber)=pixelsize*prebinning*P(5);%positiony in meters
 sigx3Dres(k)=pixelsize*prebinning*abs(P(2));%sigmax in meters
 sigy3Dres(k)=pixelsize*prebinning*abs(P(3));%sigmay in meters
 sigxerror3Dres(k)=pixelsize*prebinning*abs(sdev(k,2));%sigmaxerror in meters
 sigyerror3Dres(k)=pixelsize*prebinning*abs(sdev(k,3));%sigmayerror in meters
 sumsigxres=sumsigx+pixelsize*prebinning*abs(P(2));
 sumsigyres=sumsigy+pixelsize*prebinning*abs(P(3));
 outputres(5:5,k,basenamenumber)= odintegral;%absorption fraction, or more accurately - optical depth

 for j=1:numberofparameters
    outputres(5+j,k,basenamenumber)=P(j);%write fit information into output
end
 for jj=1:numberofparameters
    outputres(5+numberofparameters+jj,k,basenamenumber)=(ci(jj,2)-ci(jj,1))./4; %95% ci for all fit parameters will go into output file
end
 outputres(1:1,k,basenamenumber);
 outputres(2:2,k,basenamenumber);
 outputres(3:3,k,basenamenumber);
 outputres(4:4,k,basenamenumber);
 outputres(5:5,k,basenamenumber);
 %output(6:(6+2*numberofparameters),k,basenamenumber);
 %           for j=1:numberofparameters
 %               lowererror(j,k,basenamenumber)=ci(j);
%           end
 %            for j=numberofparameters+1:2*numberofparameters
 %               uppererror(j,k,basenamenumber)=ci(j); %this seems wrong
%           end
 %for j=1:5

 %    output(13+j,k,basenamenumber)=P(j);%write fit information into output

 %      lowererror(j,k,basenamenumber)=ci(j);
 %  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 if fit600==1
figure(600+k)
 subplot(2,2,1)
 surf(pixelconv*xvec,pixelconv*yvec,residualsforfitting, 'Marker', '.');
 zlabel('Optical Depth','FontSize',axisfontsize);
 xlabel('X (microns)','FontSize',axisfontsize);
 ylabel('Y (microns)','FontSize',axisfontsize);
 set(gca,'FontSize', axisfontsize);
 % view([0 90])
 caxis('auto');
 title('true residues (NOT weighted by uncertainties)','FontSize',axisfontsize)
 colorbar
 shading flat
%axis off;
grid off;
 subplot(2, 2, 2) % plot fit
 %%%Generate plot of optical density based on fitted parameters, remove distortion of data due to dividing by weight and sqrt(number of points)
 errorod(1:row*column)=1; %
 numberofpoints=ones(column,row);%
 coordmatrix=[xdata(:),ydata(:),errorod(:),numberofpoints(:)];

if becfit==0
 residualsforfittingcalcvector=BECfitfunctionerrorsslope(P, coordmatrix);%optical depth calculated from fit parameters (NOT WEIGHTED)
 elseif becfit==1
 residualsforfittingcalcvector=gaussfitfunctionerrorsslope(P, coordmatrix);%optical depth calculated from fit parameters (NOT WEIGHTED)
 end

 residualsforfittingcalcmatrix = reshape(residualsforfittingcalcvector, size(xdata));
 surf(pixelconv*xvec,pixelconv*yvec,residualsforfittingcalcmatrix, 'Marker', '.'); % Plot fit
 zlabel('Optical Depth','FontSize',axisfontsize);
 xlabel('X (microns)','FontSize',axisfontsize);
 ylabel('Y (microns)','FontSize',axisfontsize);
 set(gca,'FontSize', axisfontsize);
 text(xresidualconstants,yresidualconstants,zresidualconstants, strcat(char('Peak OD ='), num2str(P(1)), ' \pm ', num2str(sdev(k,1))),'FontSize',textsize);
 text(xresidualconstants,yresidualconstants,zresidualconstants -P(1)/4, strcat(char('\sigma_x ='), num2str(sigx3D(k)), ' \pm ', num2str(sigxerror3D(k))),'FontSize',textsize);
 text(xresidualconstants,yresidualconstants,zresidualconstants -P(1)/2, strcat(char('\sigma_y ='), num2str(sigy3D(k)), ' \pm ', num2str(sigyerror3D(k))),'FontSize',textsize);
 colorbar
 shading flat
%axis off;
grid off;
 subplot(2, 2, 3) %plot residuals
 %form smoothed residuals
 residuals=residualsforfitting-residualsforfittingcalcmatrix;
 temp=reshape(residuals,size(xdata)); %put in matrix form
 smoothresiduals=filter2(b,temp(:,:))/smoothplotsize^2; %residue smoothed a bit for fitting
 surf(pixelconv*xvec,pixelconv*yvec,temp, 'Marker', '.');
 zlabel('Optical Depth','FontSize',axisfontsize);
 xlabel('X (microns)','FontSize',axisfontsize);
 ylabel('Y (microns)','FontSize',axisfontsize);
 set(gca,'FontSize', axisfontsize);
 % view([0 90])
 caxis('auto');
 title('true residues (NOT weighted by uncertainties)','FontSize',axisfontsize)
 colorbar
 shading flat
%axis off;
grid off;
 end  %end of fit600

  absorptioncross = 6*pi*(lambda/(2*pi))^2/(1+(2*detuning/naturalwidth)^2);     % absorption cross section; used to calculate #atoms
  %numatoms = odintegral./absorptioncross;     % number of atoms obtained from the optical dpiepth
 if becfit==1
```

```
 numatoms = odintegral./absorptioncross;     % number of atoms obtained from the optical depth
elseif becfit==0
 numatoms = P(1)*abs(P(2))*abs(P(3))*pixelsize*pixelsize*(3/4)*(8*pi/15)/(absorptioncross);
end

 numatomserror = odintegralerror.*numatoms./odintegral;  % dN = N*dOD/OD
 %numatomserror = odintegralerror./absorptioncross;      % dN = dOD/absorptioncross

 %frequencies not important to non-PAS experiments now.
 %dshift = aomvector(basenamenumber)*1000;
 dcat(k)=(-.02288*(vcovoltage(k)^4)+0.497*(vcovoltage(k)^3)-3.436*(vcovoltage(k)^2)+23.93*vcovoltage(k)+169);
 %alldata(k,1)=(dshift(basenamenumber)-dsatabs(basenamenumber)+2*(dzeeman(basenamenumber)+dPAS+(catseyesign(basenamenumber)*dcat(k))))
 alldatares(k,1)=(dshift+2*(dzeeman+2*dPAS-dsatabs+(catseyesign*dcat(k))));
 alldatares(k,2)=batchvcovoltage(k,basenamenumber);
 alldatares(k,3)=odintegralerror;
 alldatares(k,4)=odintegral;
 for index=1:(size(output,1));
     alldatares(k,index+4)=outputres(index,k,basenamenumber);
end; %fit parameters and 1-sigma error for each parameter
 alldatares(k,4+size(output,1)+1)=numatomserror;
 alldatares(k,4+size(output,1)+2)=numatoms; %Number of atoms is useful
 alldatares(k,4+size(output,1)+3)=chi3Dres(k);
 alldatares(k,4+size(output,1)+4)=pixelsize*prebinning; %size parameter saved so hardware and software binning settings only set once in imagefit.
 %tempres=size(char(fileres(k,:)));
%allfilesres(k)=fileres(k);

%            figure(10)
%             subplot(2,2,1)
%             plot(imagevco(1:counter),alldatares(:,27)./10^6,'*b')
%             hold on
%             %xlabel('DAC voltage');
%             set(gca, 'FontSize', axisfontsize);
%             ylabel('Number of atoms (10^6)');
%             title(char(basename));
%             subplot(2,2,2)
%             hold on
%             plot(imagevco(1:counter),alldatares(:,10),'*b')
%             set(gca, 'FontSize', axisfontsize);
%             ylabel('Peak OD');
%             subplot(2,2,3)
%             hold on
%             plot(imagevco(1:counter),sigx3Dres(1:counter),'*b')
%             set(gca, 'FontSize', axisfontsize);
%             %xlabel('DAC voltage');
%             ylabel('RMS width X');
%             xlabel('DAC voltage');
%             %title('frequency calibration');
%             subplot(2,2,4)
%             hold on
%             plot(imagevco(1:counter),sigy3Dres(1:counter),'db')
%             set(gca, 'FontSize', axisfontsize);
%             xlabel('DAC voltage');
%             ylabel('RMS width Y');
%             hold off

 %%Calculate the geometric average of trap freqencies
 BECfraction(k)=alldatares(k,27)./(alldatares(k,27)+alldata(k,27)); %number of BEC / total number of atoms
 Tc(k) = alldata(k,30).*10.^(-9)./(1-BECfraction(k)).^(1/3);  %The BEC transition temperature
 wbar(k) = kBoltz.*Tc(k)./(0.94.*hbar.*(alldatares(k,27)+alldata(k,27)).^(1/3)); % Geometric average of angular trap freq, in rad/second
 fbar(k) = wbar(k)./(2*pi); %Geometric average of trap freq, in Hz
 ODTPowerFactor=1.122; %ODTpower = 1.122 * ODT Voltage, measure on 01/23/2010, see Lab Notebook for detail
 ODTPower(k) = ODTVoltage(k).*ODTPowerFactor;




%            figure(25)
%             plot(imagevco(1:counter),alldatares(:,27)./(alldatares(:,27)+alldata(:,27)),'*b')
%             hold on
%             figure(889) % BEC fraction vs ODT power
%               ODTPowerFactor=1.122; %ODTpower = 1.122 * ODT Voltage, measure on 01/23/2010, see Lab Notebook for detail
%               ODTPower(k) = ODTVoltage(k).*ODTPowerFactor;
%             if droptime(filecounter)==33
%                 plot(ODTPower(k),BECfraction(k)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             elseif droptime(filecounter)==22
%                 plot(ODTPower(k),BECfraction(k)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             elseif droptime(filecounter)==16
%                 plot(ODTPower(k),BECfraction(k)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             end
%             hold on
%             xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%             ylim([0 max(BECfraction)*100+5])
%             xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%             ylabel('BEC fraction [%]','FontSize',axisfontsize);
%             hold on
%             stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%             if filecounter==numberofbasenames
%                 legend(stringmatrix,'Location','Best');
%             end
%
%
%             figure(888) % fbar vs ODT power
%              ODTPowerFactor=1.122; %ODTpower = 1.122 * ODT Voltage, measure on 01/23/2010, see Lab Notebook for detail
%             if droptime(filecounter)==33
%                 plot(ODTPower(k),fbar(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             elseif droptime(filecounter)==22
%                 plot(ODTPower(k),fbar(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             elseif droptime(filecounter)==16
%                 plot(ODTPower(k),fbar(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%             end
```

```
%               hold on
%                xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%                ylim([0 max(fbar)+5])
%                xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%                ylabel('Geometric average freq [Hz]','FontSize',axisfontsize);
%                hold on
%                stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%               if filecounter==numberofbasenames
%                    legend(stringmatrix,'Location','Best');
%               end
%
%               figure(887) % sample temperature vs ODT power
%                if droptime(filecounter)==33
%                    plot(ODTPower(k),avgtemp(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%                elseif droptime(filecounter)==22
%                    plot(ODTPower(k),avgtemp(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%                elseif droptime(filecounter)==16
%                    plot(ODTPower(k),avgtemp(k),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%               end
%               hold on
%                xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%                ylim([0 max(avgtemp)+5])
%                xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%                ylabel('Samples Temperature [nKelvin]','FontSize',axisfontsize);
%               hold on
%               stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%               if filecounter==numberofbasenames
%                    legend(stringmatrix,'Location','Best');
%               end

%               figure(886) % BEC number vs ODT power
%                if droptime(filecounter)==33
%                    plot(ODTPower(k),alldatares(k,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%                   hold on
%                elseif droptime(filecounter)==22
%                    plot(ODTPower(k),alldatares(k,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%                   hold on
%                elseif droptime(filecounter)==16
%                    plot(ODTPower(k),alldatares(k,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',7)
%                   hold on
%               end
%               hold on
%                xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%                ylim([0 max(alldatares(:,27))+3000])
%                xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%                ylabel('Number of BEC','FontSize',axisfontsize);
%               hold on
%               stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%               if filecounter==numberofbasenames
%                    legend(stringmatrix,'Location','Best');
%               end




%               figure(800+k)   %Check OD of pedestal and BEC
%                nogradoddata21=nogradoddata2(:)-nogradoddata1(:);
%                nogradoddataX=1:1:length(nogradoddata21);
%                %plot(nogradoddataX,nogradoddata21,'.b')
%               hold on
%                plot(nogradoddataX,nogradoddata4(:),'.g')
%               hold on
%                plot(nogradoddataX,Z22(:),'xr')
%               hold off



            if plotresiduallinedensity==1
              figure(700+filecounter)
              xoptimal=floor(P(4));
              yoptimal=floor(P(5));
              if becfit==1
               yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*exp(-(((xoptimal-P(4)).^2)/(2*P(2)^2))-(((yvec-P(5)).^2)/(2*P(3)^2))))));%./
(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
               xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*exp(-(((xvec-P(4)).^2)/(2*P(2)^2))-(((yoptimal-P(5)).^2)/(2*P(3)^2)))));
              elseif becfit==0
               yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*heaviside(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./P(3)).^2).*(1-((xoptimal-P(4))./P(2)).^2-((yvec-P(5))./
P(3)).^2).^(3/2)));%./(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
               xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*heaviside(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./P(3)).^2).*(1-((xvec-P(4))./P(2)).^2-((yoptimal-P(5))./
P(3)).^2).^(3/2)));
              end

              subplot(2,2,k)
              plot((residualsforfitting(floor(P(5))-1,:)+residualsforfitting(floor(P(5)),:)+residualsforfitting(floor(P(5))+1,:))/3,'.b')
              hold on
              plot(xvec,xslice,'-b')
              hold on
               %plot(xvec,errorodmatrix(:,yoffset)./BECweight,'-b')
              %hold on
              plot((residualsforfitting(:,floor(P(4))-1)+residualsforfitting(:,floor(P(4)))+residualsforfitting(:,floor(P(4))+1))/3,'.r')
              hold on
               %plot(yvec,errorodmatrix(xoffset,:)./BECweight,'-r')
              %hold on
              plot(yvec,yslice,'-r')
              hold on
              xlabel('x and y position');
              ylabel('Normalized Density of Residuals');
              if k==8
                  legend('x','y','Position','EastOutside');
              end
              title(strcat(num2str(imagevco(k)),'ms'));
```

```
hold off

    %Z =((offset+slopex*(x-xoffset)+slopey*(y-yoffset)+amplitude*exp(-(((x- xoffset).^2)/(2*sigx^2))-(((y- yoffset).^2)/(2*sigy^2))))./(w*sqrtnumbpoints)); %

%                   figure(3000+filecounter)
%                    xoptimal=floor(P(4));
%                    yoptimal=floor(P(5));
%                       yslice=((P(6)+P(7)*(xoptimal-P(4))+P(8)*(yvec-P(5))+P(1)*exp(-(((xoptimal-P(4)).^2)/(2*P(2)^2))-(((yvec-P(5)).^2)/(2*P(3)^2)))));%./
(errorodmatrix(xoffset,:).*sqrt(numberofpoints(1,1)));
%                       xslice=((P(6)+P(7)*(xvec-P(4))+P(8)*(yoptimal-P(5))+P(1)*exp(-(((xvec-P(4)).^2)/(2*P(2)^2))-(((yoptimal-P(5)).^2)/(2*P(3)^2)))));
%                    subplot(4,4,k)
%                    % plot((Zguess(floor(P(5))-1,:)+Zguess(floor(P(5)),:)+Zguess(floor(P(5))+1,:))/3-xslice,'.b')
%                    plot(Zguess(floor(P(5)),:)-xslice,'.b')
%
%                    hold on
%                    %plot(xvec,xslice,'-b')
%                    hold on
%                      plot(xvec,errorodmatrix(:,yoffset)./BECweight/10,'-b')
%                    hold on
%                    %plot((Zguess(:,floor(P(4))-1)+Zguess(:,floor(P(4)))+Zguess(:,floor(P(4))+1))/3-yslice(:),'.r')
%                    plot(Zguess(:,floor(P(4)))-yslice(:),'.r')
%
%                    hold on
%                      plot(yvec,errorodmatrix(xoffset,:)./BECweight/10,'-r')
%                    hold on
%                    %plot(yvec,yslice,'-r')
%                    hold on
%                     xlabel('x and y position');
%                     ylabel('Normalized Density');
%                   if k==8
%                        legend('x','y','Position','EastOutside');
%                   end
%                     title(strcat(num2str(imagevco(k)),'ms'));
%                    hold off
  end %end of plotresiduallinedensity case structure




end %end if fitresiduals==1




 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 end % end for skip fit if fit ==0
 %allfiles(k,1:temp(2))=char(file(k,:));



end %loop through files in batch
%get data from batch file
%
%              figure(889) % BEC fraction vs ODT power
%               if droptime(filecounter)==33
%                   plot(ODTPower(:),BECfraction(:)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==22
%                   plot(ODTPower(:),BECfraction(:)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==16
%                   plot(ODTPower(:),BECfraction(:)*100,MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%              end
%               hold on
%               xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%               ylim([0 max(BECfraction)*100+5])
%               xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%               ylabel('BEC fraction [%]','FontSize',axisfontsize);
%               hold on
%               stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%              if filecounter==numberofbasenames
%                   legend(stringmatrix,'Location','Best');
%              end
%
%
%              figure(888) % fbar vs ODT power
%               ODTPowerFactor=1.122; %ODTpower = 1.122 * ODT Voltage, measure on 01/23/2010, see Lab Notebook for detail
%               if droptime(filecounter)==33
%                   plot(ODTPower(:),fbar(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==22
%                   plot(ODTPower(:),fbar(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==16
%                   plot(ODTPower(:),fbar(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%              end
%               hold on
%               xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%               ylim([0 max(fbar)+5])
%               xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%               ylabel('Geometric average freq [Hz]','FontSize',axisfontsize);
%               hold on
%               stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%              if filecounter==numberofbasenames
%                   legend(stringmatrix,'Location','Best');
%              end
%
%              figure(887) % sample temperature vs ODT power
%               if droptime(filecounter)==33
%                   plot(ODTPower(:),avgtemp(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==22
%                   plot(ODTPower(:),avgtemp(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               elseif droptime(filecounter)==16
```

```
%                 plot(ODTPower(:),avgtemp(:),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%             end
%             hold on
%              xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%              ylim([0 max(avgtemp)+5])
%              xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%              ylabel('Samples Temperature [nKelvin]','FontSize',axisfontsize);
%             hold on
%             stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%            if filecounter==numberofbasenames
%                legend(stringmatrix,'Location','Best');
%            end
%
%            figure(886) % BEC number vs ODT power
%             if droptime(filecounter)==33
%                plot(ODTPower(:),alldatares(:,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               hold on
%             elseif droptime(filecounter)==22
%                plot(ODTPower(:),alldatares(:,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               hold on
%             elseif droptime(filecounter)==16
%                plot(ODTPower(:),alldatares(:,27),MARKERS{basenamenumber},'Color',COLORS(basenamenumber,:),'MarkerSize',5)
%               hold on
%            end
%             hold on
%              xlim([min(ODTPower)-0.01 max(ODTPower)+0.01])
%              ylim([0 max(alldatares(:,27))+3000])
%              xlabel('ODT Power [Watt]','FontSize',axisfontsize);
%              ylabel('Number of BEC','FontSize',axisfontsize);
%            hold on
%             stringmatrix{filecounter}=strcat(num2str(droptime(basenamenumber)),' ms drop time');
%            if filecounter==numberofbasenames
%                legend(stringmatrix,'Location','Best');
%            end

end %end loop through batches

%% Wrap Up
status = 'done'
```

The second program is $FugacityToReducedTempWithError.nb$, used to read in values of fugacity, calculate T/T_F and write the values to an output file

Calculate T/Tf from fugacity

In[1]:=
```
BatchDirectory =
  "/Volumes/strontium86/GROUP PAPERS/2010MixedGasDegeneracy/T-Tf vs Time Figure/";
BaseName = "MixExpODT8787Evap22msdrop3520";
DataFile = BatchDirectory <> BaseName <> "fugacity.txt";


reducedTemperature[fugacity_] := (-1 / (6 * PolyLog[3, -fugacity]))^(1 / 3);
data = Import[DataFile, "Table"]
For[i = 1, i < Dimensions[data][[1]] + 1, i++,
 data[[i, 2]] = N[reducedTemperature[data[[i, 2]]]];
 data[[i, 3]] = N[reducedTemperature[data[[i, 3]]]];
 data[[i, 4]] = N[reducedTemperature[data[[i, 4]]]]]
ListPlot[data]
data
Export[BatchDirectory <> BaseName <> "reducedtemp.dat", data]
```

[1] D. A. Butts and D. S. Rokhsar, Phys. Rev. A **55**, 4346 (Jun 1997), http://link.aps.org/doi/10.1103/PhysRevA.55.4346.

[2] B. DeMarco, *Quantum Behavior of an Atomic Fermi Gas*, Ph.D. thesis, University of Colorado (2001).

[3] P. G. Mickelson, *Trapping and Evaporation of* ^{87}*Sr and* ^{88}*Sr Mixtures*, Ph.D. thesis, Rice University (2010).

[4] M. Yan, R. Chakraborty, A. Mazurenko, P. G. Mickelson, Y. N. M. de Escobar, B. J. DeSalvo, and T. C. Killian, Phys. Rev. A **83**, 032705 (Mar 2011), http://link.aps.org/doi/10.1103/PhysRevA.83.032705.

[5] S. Giorgini, L. Pitaevskii, and S. Stringari, Rev. Mod. Phys. **80**, 1215 (2008).