A PRECISION MEASUREMENT OF THE FINE STRUCTURE CONSTANT

a dissertation
submitted to the department of physics
and the committee on graduate studies
of stanford university
in partial fulfillment of the requirements
for the degree of
doctor of philosophy

Joel Moses Hensley
August 2001

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Steve Chu
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Blas Cabrera

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Stephen E. Harris

Approved for the University Committee on Graduate Studies:

Abstract

Using an atom interferometer method based on adiabatic transfer between atomic states, we measure the recoil frequency shift $f_{\mathrm{rec}} = h/(m_{\mathrm{Cs}}\lambda_{\mathrm{eff}}^2)$ of cesium due to the absorption of a photon of effective inverse wavelength $1/\lambda_{\mathrm{eff}} = 1/\lambda_{33^0} + 1/\lambda_{43^0}$, where λ_{33^0} and λ_{43^0} are the wavelengths of the D1 transitions between the $F=3$ and $F=4$ hyperfine ground states of the $6S_{1/2}$ energy level and the $F'=3$ hyperfine $6P_{1/2}$ excited state, respectively. We report a value of $f_{\mathrm{rec}} = 15\,006.276\,9996(874)$ Hz, where the single standard deviation uncertainty includes both the systematic and statistical uncertainties after averaging more than 2 800 data points. With independent measurements of the Rydberg constant, the proton to electron mass ratio, the cesium to proton mass ratio, and the wavelengths for the D1 transition of cesium, we derive a value for the fine structure constant $\alpha^{-1} = 137.035\,999\,710(427)(401)$, where the first error bar is the combined uncertainty, equivalent to a fractional error of 3.1×10^{-9}, from both this and the above mentioned independent measurements and the second value is the contribution to the uncertainty from just this work.

"... One thirty-seven is Eddington's fine structure constant, of course, and it turns up over and over in nuclear physics. But it is more than that. Suppose you take the inverse, that is one over one thirty-seven, and express it as a decimal. The first three digits are Double Ought Seven, James Bond's identification as a killer. There is the lethality of the universe for you! The first eight digits are Clarke's Palindrome, point oh oh seven two nine nine two seven oh. There is its symmetry. Deadly, and two-faced, that is the fine structure constant! Or," he mused, "perhaps I should say, there is its inverse. Which would imply that the universe itself is the inverse of that? Namely kind and uneven? ..."

—*Frederik Pohl*, 1980
Beyond the Blue Event Horizon

Acknowledgments

I would be remiss not to thank my advisor Steve Chu for providing me the opportunity to work on such a troublesome and difficult (I mean, stimulating and challenging) experiment. More importantly, I would like to thank the people who taught me everything I know. My bad-ass physicist (BAP) father never had the guts to steer me away from this thankless field. I will always be grateful to Duke University Physics Department faculty member Frank DeLucia (now at Ohio State University) who saw the taint of the dark side in me and started me down this long and twisted road. Duke physics professors Hugh Robinson, John Thomas, and Dan Gauthier, Columbia University professor George Flynn, NASA research fellow Sheldon Green, and Lawrence Livermore National Laboratory Staff Scientist Nicolas J. Collela were also instrumental. At Stanford, my first role model was senior graduate student Brent Young, from whom I inherited in addition to the nightmare we know of as $\hbar/M_{\mathrm{Cs}}$, a fascination with electronics and a preoccupation to detail. Other graduate students of varying sizes, shapes, and temperaments who also taught me a great deal include (but are not limited to) Achim Peters (Mr. Vibration), Heun-Jin Lee (Chief Cynic), and Todd Gustavson (Toddbert). More recently, I would also like to thank fellow salt miners Keng Yeow Chung, Cheng Chen, Jamie Kerman, and Hazen Babcock for a great deal of commiseration and assistance, both technical and theoretical. I also had the extremely pleasant experience of working and sharing this experiment with graduate students Richard Swartz and Kurt Franke. For the last two years, I have also spent a great deal of positive time working with post-doc Andreas Wicht. Except for brief moments when he was recovering from a "four-by-four", he was always more than willing to help me understand and work through both work-related and non-work-related challenges. I will always be indebted to him for his assistance and for taking much of the heat. There are many other wonderful people I had the privilege of getting to know at Stanford, many of whom I will forget to mention. Ken Sherwin, the eternal spirit of the Varian Building and Yoda-figure to all of its temporary occupants. I had more in common with him than I will ever admit. Marcia Keating, first adoptive mother to us all. Rosenna Yau, for her consistent and rock-steady support and guidance. Barbara Heather for understanding. Stewart Kramer for helping us through all of the Varian building's growing and aging pains. Machinists Wolfgang Jung, Karlheinz Merkle, Matt Chuck, John Kirk, and Mehmet Solyali for tactfully averting their eyes or patiently lending a hand with my amateur attempts to make the machines behave. I would also like to thank Professor Mark Kasevich (now at Yale University) and Barry Taylor for their professional guidance. Barry Taylor, more than any other, inspired me to try to make the best measurement I could make.

I acknowledge the support of an Air Force graduate fellowship and grants from AFOSR and NSF.

Finally, and most importantly, I would like to thank the members of my ever expanding family for their support and patience. Both my son, Ethan, who mercifully wont remember any of this, and my wife, Betsy, who will remember *all* of it, deserve much better but never asked for anything more.

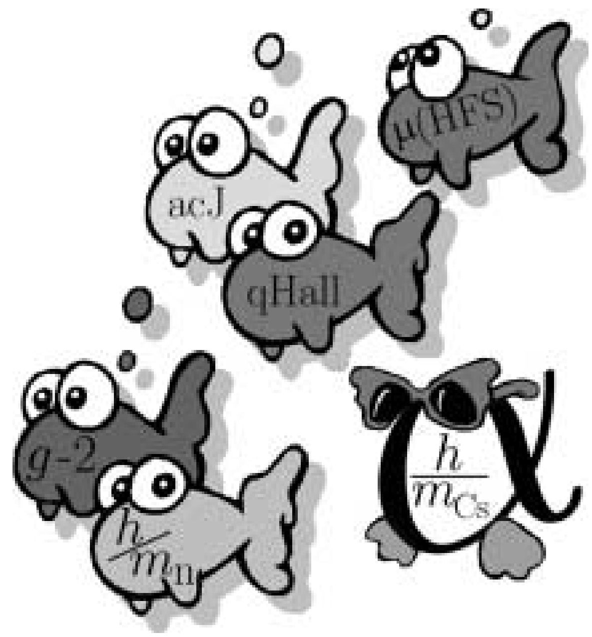
Contents

List of Tables

List of Figures

Chapter 1

Introduction



The fine structure constant α is a dimensionless number that describes the strength of the electromagnetic interaction between matter and light. It was originally introduced by Sommerfeld in 1916 to describe the size of the relativistic correction, termed "fine structure", to the energies levels of the Bohr hydrogen atom [1]. Because it appears any time electromagnetic interactions are involved, it links almost all disciplines of physics from elementary particle to macroscopic systems. As a result, throughout all of physics there have been several fundamentally different approaches to determining its value [2]. Currently, the five most precise determinations of α are based on the quantum Hall effect, the electron's anomalous magnetic moment, the ac Josephson effect, the muonium hyperfine structure, and the measurement of h/m for slow neutrons. The values from these measurements used in the most recent statistical combination of all of the fundamental constants [3] are shown together with the value from this work in Figure 1.1. The quantum Hall approach measures the Hall resistance R_{H} and thus α directly, but it is limited to an accuracy of 19.7 parts per billion (ppb) by the uncertainty in the calibration of the standard Ohm. Determining α from a measurement of a_{e} [4], the anomalous magnetic moment of the electron, requires expanding a_{e} in powers of α and calculating the coefficients of this expansion. These calculations, which require an increasing complicated application of quantum electrodynamics (QED) theory, have been completed through the α^3 term and most of the α^4 term. Assuming the theory is correct, this determination of α will soon be limited only by the 3.4 ppb uncertainty in the determining a_e. The current discrepancy between the standard model and the muon anomalous magnetic moment a_μ is ~ 4 ppb. Because of the electron's smaller mass, if this discrepancy is real, it would appear at approximately the 10^{-13} level for a_e, or approximately 0.5 ppb in α [5]. Therefore, if the 1987 measurement of a_e [6] could be significantly improved, comparing the resulting theory-dependent value of α with another measurement that does not depend as heavily on QED would represent a significant test of the standard model. The 3 ppb measurement of α presented here is only a factor of six away from this limit.

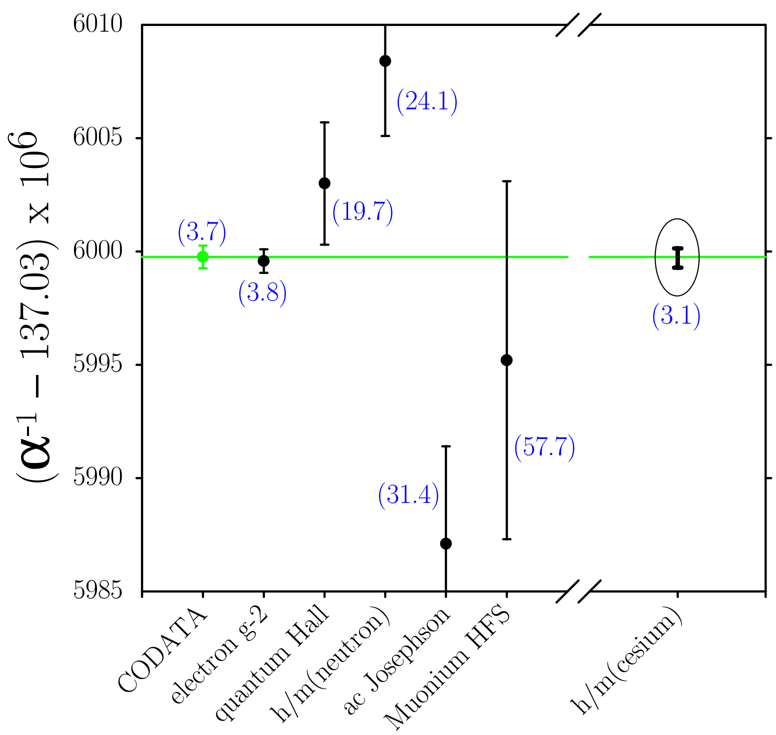


Figure 1.1: **The current determinations of the fine structure constant α.** The value from the most recent adjustment of all the fundamental constants is shown in gray. This value is primarily determined by the five most precise measurements of α, based on the anomalous magnetic moment of the electron $(g-2)$, the quantum Hall effect, neutron interferometry, the ac Josephson effect, and the muonium hyperfine structure. Our work will discuss the details behind and the results of another determination of α based on a measurement of the photon recoil frequency for cesium. The values in parentheses () are the fractional single-standard deviation uncertainties in parts per billion.

The remaining methods including the one presented here determine α indirectly by combining one or more precision measurements with the value of the Rydberg constant

$$R_\infty = \frac{m_e c}{2h}\alpha^2 \tag{1.1}$$

measured to an uncertainty of 0.0076 ppb [3].

The ac Josephson technique first determines e/h, the elementary charge over Planck's constant, by carefully measuring the frequency of current oscillations produced when a known voltage is applied to a superconducting junction. The electron mass m_e in equation (1.1) can be expressed in terms of the Bohr magneton μ_B, the proton's magnetic moment μ_{p^0} in water, and the corresponding gyromagnetic moment γ_{p^0}, all of which can be accurately measured. Combining these measurements produces a value for α whose 31.4 ppb accuracy is currently limited by the experimental uncertainty in determining γ_{p^0}.

One can also determine α by comparing the measured ground state hyperfine splitting Δf_μ of muonium with the theoretically predicted value that depends on R_∞, the muon electron mass ratio m_μ/m_e, and α. Currently the 58 ppb uncertainty in α from this technique is limited mostly by the experimental determination of the muon mass. However, even if the mass ratio were known more precisely, complexities in the theory for predicting Δf_μ will limit the uncertainty to $\sim$17 ppb for quite some time.

Finally, with a precise value for R_∞, one of the most basic methods for experimentally determining α is to measure h/m, the ratio of Planck's constant to the mass m of some particle. Combining this result with a measurement of m/m_e allows one to determine α from equation (1.1). By scattering a beam of neutrons off an ultra-pure silicon crystal, the velocity and de Broglie wavelength of the neutron and thus the quantity h/m_n have been measured to an accuracy of 24.1 ppb, which is large compared to the 2.2 ppb uncertainty in the neutron to electron mass ratio m_n/m_e. This uncertainty in the measured value of h/m_n comes primarily from not knowing the exact effective lattice spacing of the silicon crystal. Even with a more exact x-ray measurement of the crystal lattice spacing, however, it is not clear that all effects such as impurities and mechanical stresses of the silicon crystal can be sufficiently controlled and/or characterized.

In a completely different way, we perform an analogous measurement with cesium atoms. Instead of scattering the cesium atoms off a physical object, we use a laser with well-defined wavelength λ to impart a momentum $p = h/\lambda$ to a cesium atom. In order to conserve momentum, the atom will recoil with velocity $\Delta v = p/m_{\mathrm{Cs}} = 1/\lambda(h/m_{\mathrm{Cs}})$. According to the first-order Doppler effect, this change Δv in the atom's velocity shifts the perceived value of the atom's internal resonances by a frequency $\Delta f = \Delta v/\lambda = 1/\lambda^2(h/m_{\mathrm{Cs}})$. If we know the wavelength λ of our laser accurately and we measure this frequency shift, often called the recoil shift f_{rec}, we can determine a value for h/m_{Cs}, from which we can derive a value for α according to

$$\begin{aligned} \alpha^2 &= \frac{2R_\infty}{c}\frac{m_{\mathrm{p}}}{m_{\mathrm{e}}}\frac{m_{\mathrm{Cs}}}{m_{\mathrm{p}}}\frac{h}{m_{\mathrm{Cs}}} \\ &= \frac{2R_\infty}{c}\frac{m_{\mathrm{p}}}{m_{\mathrm{e}}}\frac{m_{\mathrm{Cs}}}{m_{\mathrm{p}}} f_{\mathrm{rec}}\lambda^2 \end{aligned} \tag{1.2}$$

Note that in addition to a value for R_∞, this approach requires measurements of the proton to electron and the cesium to proton mass ratios, and the wavelength of the atomic transitions we lock our lasers to. Fortunately, the proton to electron mass ratio has been measured by Van Dyck and Schwinberg with an uncertainty of 2.1 ppb [7], and a beautiful measurement performed by the group of D. Pritchard assigns a value to the cesium mass with an uncertainty of 0.2 ppb. Finally, the group of T. Haensch has determined the frequency of the cesium D1 transition to an accuracy of 0.13 ppb. Since we lock our lasers to this transition, this frequency measurement determines the value of λ and thus the size of the recoil. And, since all of the quantities in equation (1.2) are either defined or measured to the ppb level, a precise measurement of the recoil shift $f_{\mathrm{rec}} = h/m_{\mathrm{Cs}}(1/\lambda^2)$ for cesium will lead to a measurement of α with an uncertainty of only a few parts in one billion.

1.1 Photon recoil measurement

The first measurement of the recoil frequency shift was made in an heroic experiment by Hall, Bordé and Uehara [8], achieving a resolution $\Delta f_{\mathrm{rec}}/f_{\mathrm{rec}} = 2.3 \times 10^{-3}$ using a laser with a linewidth of 200 Hz, 32 cm diameter optics and an absorption cell with a 13 m path length. Systematic effects led to a 6×10^{-3} discrepancy from the known value of h/M.

To look more closely at the definition of the recoil frequency, consider an atom of mass m moving with velocity $\mathbf{v}_{\mathrm{i}}$. For now, consider just two of the internal electronic states of the atom: $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$. These states have energies $E_{\mathrm{a}} = \hbar\omega_{\mathrm{a}}$ and $E_{\mathrm{b}} = \hbar\omega_{\mathrm{b}}$, respectively. A laser with frequency $\omega_{\mathrm{L}} \simeq \omega_{\mathrm{b}} - \omega_{\mathrm{a}} \equiv \omega_{\mathrm{ab}}$ resonantly drives an electric dipole transition between these two states. A photon from this laser field has momentum $\hbar\mathbf{k}$, where $k = |\mathbf{k}| = \omega_{\mathrm{L}}/c$ is the laser's wavenumber. Assume the atom starts in state $|\mathrm{a}\rangle$, absorbs a photon from the laser field, and ends in state $|\mathrm{b}\rangle$. Before the absorption of this single photon the total energy and momentum of the system are

$$E_{\mathrm{initial}} = \tfrac{1}{2}m|\mathbf{v}_{\mathrm{i}}|^2 + \hbar\omega_{\mathrm{a}} + \hbar\omega_{\mathrm{L}} \tag{1.3}$$

$$\mathbf{p}_{\mathrm{initial}} = m\mathbf{v}_{\mathrm{i}} + \hbar\mathbf{k} \tag{1.4}$$

When the atom absorbs the photon it also absorbs the photon's momentum and recoils with velocity change $\Delta\mathbf{v}$ in the direction of the photon. The energy and momentum are now

$$E_{\mathrm{final}} = \tfrac{1}{2}m|\mathbf{v}_{\mathrm{i}} + \Delta\mathbf{v}|^2 + \hbar\omega_{\mathrm{b}} \tag{1.5}$$

$$\mathbf{p}_{\mathrm{final}} = m(\mathbf{v}_{\mathrm{i}} + \Delta\mathbf{v}) \tag{1.6}$$

In order to conserve momentum $\mathbf{p}_{\mathrm{final}} = \mathbf{p}_{\mathrm{initial}}$, the atom's final velocity $\mathbf{v}_{\mathrm{f}}$ must be

$$\mathbf{v}_{\mathrm{f}} = \mathbf{v}_{\mathrm{i}} + \Delta\mathbf{v} = \mathbf{v}_{\mathrm{i}} + \frac{\hbar}{m}\mathbf{k} \tag{1.7}$$

Similarly, the change in energy is

$$
\begin{aligned}
&E_{\text{final}} - E_{\text{initial}} \\
&\quad= \tfrac{1}{2}m|\mathbf{v}_{\text{i}} + (\hbar/m)\mathbf{k}|^2 + \hbar\omega_{\text{b}} - \tfrac{1}{2}m|\mathbf{v}_{\text{i}}|^2 - \hbar\omega_{\text{a}} - \hbar\omega_{\text{L}} \\
&\quad= \tfrac{1}{2}m\left[v_{\text{i}}^2 + 2(\hbar/m)\mathbf{v}_{\text{i}}\cdot\mathbf{k} + (\hbar/m)^2k^2\right] - \tfrac{1}{2}mv_{\text{i}}^2 + \hbar(\omega_{\text{b}} - \omega_{\text{a}}) - \hbar\omega_{\text{L}} \\
&\quad= \hbar(\mathbf{v}_{\text{i}}\cdot\mathbf{k}) + \frac{(\hbar k)^2}{2m} + \hbar\omega_{\text{ab}} - \hbar\omega_{\text{L}} \qquad (1.8)
\end{aligned}
$$

To conserve energy, the laser frequency ω_{L} must be

$$\omega_{\text{L}} - \omega_{\text{ab}} = \mathbf{v}_{\text{i}}\cdot\mathbf{k} + \frac{\hbar k^2}{2m} \qquad (1.9)$$

The first term on the right hand side of equation (1.9) is the first order Doppler shift. The second term is the recoil shift[1] $\frac{1}{2}\omega_{\text{rec}} = \frac{1}{2}(2\pi f_{\text{rec}})$.

$$f_{\text{rec}} = \frac{1}{2\pi}\frac{\hbar k^2}{m} = \frac{1}{\lambda^2}\frac{h}{m} \qquad (1.10)$$

where $\lambda = 2\pi/k$ is the laser's wavelength. This term represents the amount of energy that must be added to the photon energy in order to compensate for the change of the atom's kinetic energy when it recoils with the photon's momentum. Once the atom absorbs a photon and changes its internal state, it can no longer absorb any more photons. Via its interaction with the laser field, it can however undergo a stimulated emission process, whereby it emits a photon of momentum $\hbar\mathbf{k}$ with the laser field, recoils in the opposite direction, and returns to state $|\text{a}\rangle$. In this stimulated emission case, the initial and final energy and momentum are

$$E_{\text{initial}} = \tfrac{1}{2}m|\mathbf{v}_{\text{i}}|^2 + \hbar\omega_{\text{b}} \quad\Big|\quad E_{\text{final}} = \tfrac{1}{2}m|\mathbf{v}_{\text{i}} + \Delta\mathbf{v}|^2 + \hbar\omega_{\text{a}} + \hbar\omega_{\text{L}} \qquad (1.11)$$

$$\mathbf{p}_{\text{initial}} = m\mathbf{v}_{\text{i}} \quad\Big|\quad \mathbf{p}_{\text{final}} = m(\mathbf{v}_{\text{i}} + \Delta\mathbf{v}) + \hbar\mathbf{k} \qquad (1.12)$$

[1] The recoil shift is often defined to include the $\frac{1}{2}$, and sometimes with the $\frac{1}{2}$ replaced with a 2 (see [9]). Throughout this work, we will use the definition given in equation (1.10).

Again, to conserve momentum we must have

$$\Delta \mathbf{v} = -\frac{\hbar}{m}\mathbf{k} \tag{1.13}$$

which implies that the change in energy is

$$\begin{aligned}
&E_{\text{final}} - E_{\text{initial}} \\
&= \tfrac{1}{2}m|\mathbf{v}_{\text{i}} - (\hbar/m)\mathbf{k}|^2 + \hbar\omega_{\text{a}} + \hbar\omega_{\text{L}} - \tfrac{1}{2}m|\mathbf{v}_{\text{i}}|^2 - \hbar\omega_{\text{b}} \\
&= \tfrac{1}{2}m\left[v_{\text{i}}^2 - 2(\hbar/m)\mathbf{v}_{\text{i}}\cdot\mathbf{k} + (\hbar/m)^2k^2\right] - \tfrac{1}{2}mv_{\text{i}}^2 - \hbar(\omega_{\text{b}} - \omega_{\text{a}}) + \hbar\omega_{\text{L}} \\
&= -\hbar(\mathbf{v}_{\text{i}}\cdot\mathbf{k}) + \frac{(\hbar k)^2}{2m} - \hbar\omega_{\text{ab}} + \hbar\omega_{\text{L}}
\end{aligned} \tag{1.14}$$

To conserve energy, the laser frequency is exactly the same as equation (1.9) except the last term is negative. For stimulated absorption or emission, the resonance condition for the laser is thus

$$\omega_{\text{L}} - \omega_{\text{ab}} = \mathbf{v}_{\text{i}}\cdot\mathbf{k} \pm \tfrac{1}{2}\omega_{\text{rec}} \tag{1.15}$$

where the recoil shift term $\omega_{\text{rec}} = \hbar k^2/m$ is positive for absorption and negative for emission.

We are now ready to propose a direct experiment for measuring the value of the recoil shift given in equation (1.10). Once again, assume the atom starts in $|\text{a}\rangle$ with velocity v_1 along the laser beam direction. At some time we expose the atom to laser light for a finite amount of time. The laser has wavevector $|\mathbf{k}_{\text{P1}}| = k$ and frequency ω_{P1}, which we set near resonance. From equation (1.15), $\omega_{\text{P1}} = \omega_{\text{ab}} + \mathbf{v}_1\cdot\mathbf{k}_{\text{P1}} + \frac{1}{2}\omega_{\text{rec}}$. We control the duration and intensity of this light pulse so that the atom is transfered from $|\text{a}\rangle$ to a superposition of states $|\text{a}\rangle$ and $|\text{b}\rangle$. In order to go from state $|\text{a}\rangle$ to $|\text{b}\rangle$ the atom must absorb a photon from the laser field and recoil with velocity $v_{\text{r}} = \hbar k/m$ in the direction of the laser. As a result, the parts of the atom in states $|\text{a}\rangle$ and $|\text{b}\rangle$ have velocities that differ by v_{r}. As depicted in Figure 1.2a, these two components of the atomic state begin to separate in space. At some later time we again pulse the laser light on and off, but for this pulse we reverse the direction of the laser beam, so that $\mathbf{k}_{\text{P2}} = -\mathbf{k}_{\text{P1}}$. Finally, we detect the atomic state and repeat the measurement to

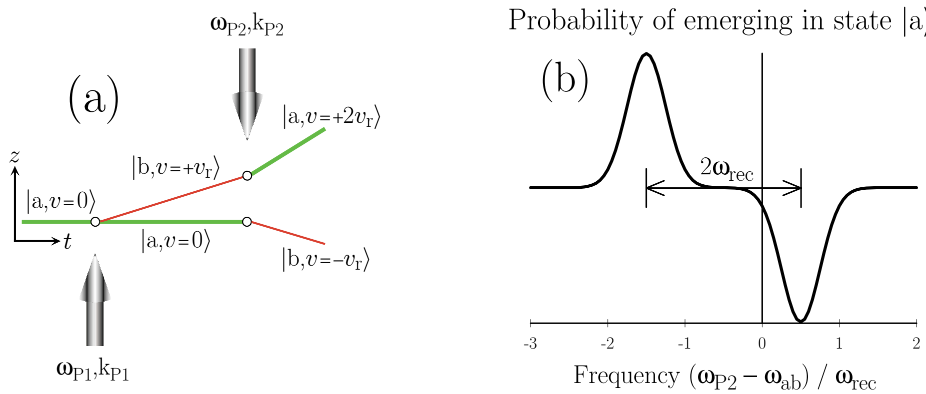


Figure 1.2: **The fundamental measurement of the recoil frequency shift ω_{rec}.** A two-level atom of mass m initially in state $|\text{a}\rangle$ interacts with a laser field of frequency ω_{P1} and wavenumber k_{P1} such that the atom is transfered to an equal superposition of states $|\text{a}\rangle$ and $|\text{b}\rangle$. In order to go from state $|\text{a}\rangle$ to state $|\text{b}\rangle$, the atom must absorb a photon of momentum $\hbar k_{\text{P1}}$ and recoil with velocity change $v_{\text{r}} = \hbar k_{\text{P1}}/m$. As shown in (a), where for the moment we have assumed that the atom starts at rest, the part of the atomic state projecting onto $|\text{b}\rangle$ separates spatially from the part of the atomic state projecting onto $|\text{a}\rangle$. If at some time later, we reverse the direction of our laser and again illuminate the atom, but this time scan the laser's frequency ω_{P2} and repeat the entire measurement to determine the probability of emerging in state $|\text{a}\rangle$, we see the lineshape shown in (b). Far off-resonance the laser does not address the atom, so it will emerge in state $|\text{a}\rangle$ with probability 0.5. When the laser is resonant with the part of the atomic state in $|\text{b}\rangle$, it drives the atom from $|\text{b}\rangle$ to $|\text{a}\rangle$ and thus *increases* the probability of emerging in $|\text{a}\rangle$. Similarly, when the laser is resonant with the part of the atomic state in $|\text{a}\rangle$, it drives the atom out of $|\text{a}\rangle$ into $|\text{b}\rangle$ and this *reduces* the probability of emerging in $|\text{a}\rangle$. The frequency separation between these two resonance is twice the recoil frequency $2\omega_{\text{rec}}$, independent of the atom's initial velocity.

determine the probability of emerging in state $|\text{a}\rangle$. This probability as a function of the frequency ω_{P2} of the second laser pulse is shown in Figure 1.2b. Two resonance features are present, one for each part of the atomic state. When the second laser pulse is resonant with the $|\text{a}\rangle$ part of the atomic state at frequency

$$\begin{aligned}\omega_{\text{P2}} = \omega_{\text{P2}}(|\text{a}\rangle) &= \omega_{\text{ab}} + \mathbf{v}_1 \cdot \mathbf{k}_{\text{P2}} + \tfrac{1}{2}\omega_{\text{rec}} \\ &= \omega_{\text{ab}} - v_1 k + \frac{\hbar k^2}{2m} \end{aligned} \tag{1.16}$$

it transfers the atom from $|\text{a}\rangle$ to $|\text{b}\rangle$ thus decreasing the probability of it emerging in state $|\text{a}\rangle$. On the other hand, when the laser is resonant with the $|\text{b}\rangle$ part of the atomic state which is moving with velocity $v_1 + v_\mathrm{r}$, its frequency must be

$$\begin{aligned} \omega_{\mathrm{P2}} = \omega_{\mathrm{P2}}(|\mathrm{b}\rangle) &= \omega_{\mathrm{ab}} + (\mathbf{v}_1 + \frac{\hbar}{m}\mathbf{k}) \cdot \mathbf{k}_{\mathrm{P2}} - \tfrac{1}{2}\omega_{\mathrm{rec}} \\ &= \omega_{\mathrm{ab}} - v_1 k - \frac{\hbar k^2}{m} - \frac{\hbar k^2}{2m} \\ &= \omega_{\mathrm{ab}} - v_1 k - \frac{3\hbar k^2}{2m} \end{aligned} \tag{1.17}$$

and it will transfer the atom from $|\mathrm{b}\rangle$ to $|\mathrm{a}\rangle$ thus increasing the probability of emerging in state $|\mathrm{a}\rangle$. The difference between these two resonances is

$$\omega_{\mathrm{P2}}(|\mathrm{a}\rangle) - \omega_{\mathrm{P2}}(|\mathrm{b}\rangle) = 2\,\frac{\hbar k^2}{m} = 2\,\omega_{\mathrm{rec}} \tag{1.18}$$

independent of the Doppler shift from the non-zero initial velocity. For a given beam direction the atom plus laser form a closed system with energy being transferred to and from the atom for each stimulated absorption and emission process. In other words, if we do not reverse the beam direction, the first and second pulses will have the exact same apparent resonance frequency. Thus, in order to observe the recoil shift, one must reverse the beam direction.

For the cesium atoms we use in this measurement, with the laser tuned to the D1 line at 894.6 nm, this frequency difference is $2\omega_\mathrm{r} \simeq (2\pi)7.5$ kHz. Therefore, to measure h/m_{Cs} with an accuracy of one part in 10^9, we must determine the center of these resonances to within 7.5 μHz. To make matters more interesting, in a real experiment we use a sample of many atoms that has a distribution of velocities. This velocity distribution Doppler broadens the resonances. Other sources of broadening include the laser linewidth and the natural linewidth of the transition, which is $\sim$4.6 MHz for the cesium D1 transitions.

Two-photon transitions

To make this fundamental recoil measurement feasible, the first improvement we make is to replace the single laser field with two counter-propagating lasers with frequencies ω_1 and ω_2 and wavevectors $\mathbf{k}_1$ and $\mathbf{k}_2 \simeq -\mathbf{k}_1$. We derive two extremely important benefits when the atom exchanges photons with both light fields. First, we can now drive two-photon transitions between meta-stable ground states. With $|\mathrm{b}\rangle$ representing a long-lived ground state, we are not limited by the natural linewidth of a relatively short-lived excited state. Second, the stability requirements for the laser's absolute frequency are much less stringent. Because the initial and final states are separated only by the ground state hyperfine splitting (~ 9.2 GHz for cesium, see Figure 2.6), the two-photon resonance is determined by the frequency difference $\omega_1 - \omega_2$ of the two lasers. Since the difference frequency is in the microwave regime, it can easily be controlled with virtually arbitrarily fine resolution. Finally, by using two-photon transitions we further benefit by doubling the size of the recoil. Because the two lasers counter propagate, the atom's momentum changes by $\hbar k_1$ when it absorbs a photon from the first laser field and then by $-\hbar(-k_2) = \hbar k_2$ when it emits a photon into the second field directed in the opposite direction. The net change of momentum is thus $\hbar(k_1 + k_2) = \hbar k_{\mathrm{eff}}$. Similarly, the recoil frequency shift $f_{\mathrm{rec}} = (1/\lambda_1 + 1/\lambda_2)^2 h/m = h/(m\lambda_{\mathrm{eff}}^2)$ is now four times larger.

Interferometry

By extending each of the two paths in Figure 1.2a into interferometers, we can dramatically improve our resolution. Figure 1.3a shows the two interferometry geometries, originally proposed by Bordé [10, 11], each constructed with four $\pi/2$-pulses. The enclosed phase space area and thus the final phase difference between the two paths of each interferometer is proportional to the size of the recoil (see Section 2.1 for details). As we scan the frequencies ω_{P3} and ω_{P4} of the final two $\pi/2$-pulses, the phase difference between the interferometer paths varies and we observe interference fringes superimposed on the original two resonance lineshapes, as represented in Figure 1.3b. The frequency period of these fringes can be made arbitrarily small, limited only by the finite interaction time with the atoms. By superimposing fringes with linewidths as small as ~ 4 Hz on top of the much broader resonance lineshape, we effectively reduce the width of the resonances by almost four orders of magnitude. In addition, because the area of the interferometers is independent of an atom's initial velocity, all atoms contribute equally to the final signal. In order to improve the

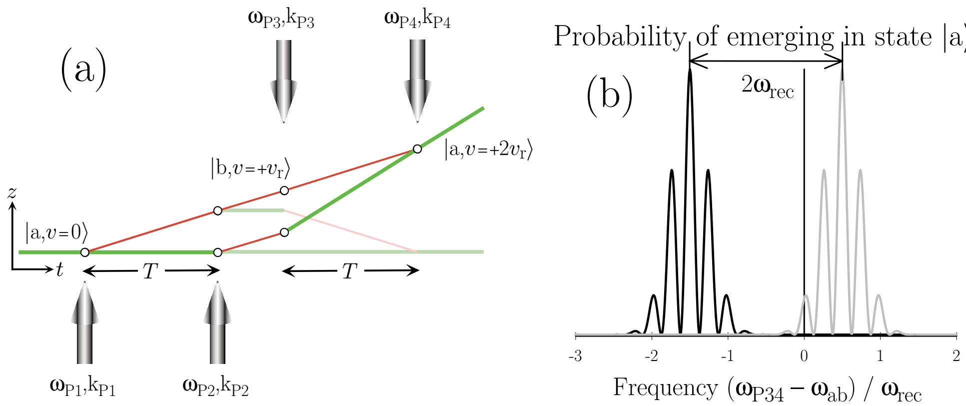


Figure 1.3: **Recoil measurement using interferometry.** Each of the two paths in Figure 1.2a are extended into an interferometer constructed with four $\pi/2$-pulses. These two interferometers superimpose fringes on the two resonances shown in Figure 1.2b. The interference fringes have frequency period $1/T$, where T is the time between the $\pi/2$-pulse pairs during which the atoms freely evolve in a superposition state. By increasing this time T, the fringe features can be made finer and finer, allowing us to more precisely find the center of the resonances. The position of the fringes and the area enclosed by each interferometer is proportional to the size of the photon recoil we are trying to measure. Since this enclosed area is independent of the atom's initial velocity, all of the atoms produce the same final phase. Without the fringes, the measurement precision would be limited by the width of the resonances which are Doppler broadened by the atoms' initial velocity distribution. In order to make a precise measurement without interferometry, we would have to dramatically reduce our signal size by selecting a small fraction of the atoms in a narrow velocity class. By observing the interference phase, however, we are no longer limited by the Doppler width and can therefore use the entire sample of atoms to make a measurement of ω_{rec}.

resolution without the interferometers, we would have to reduce the Doppler width of the resonances by selecting a particular fraction of the atoms' velocity distribution and thereby dramatically reducing the signal size. With the interferometers, we have the benefit of a large final signal without sacrificing resolution.

Additional recoils

A final resolution enhancement comes from inserting a number N of π-pulses between the second and third $\pi/2$-pulses of each interferometer. These π-pulses increase the separation between the two interferometers by $2N$ recoils, which shifts the resonances in Figure 1.3b apart by exactly $2N\omega_{\text{rec}}$. We have demonstrated interferometers with up to 50 π-pulses, in which case the resonances are separated by 102 two-photon recoils, or 204 single-photon recoils. Since the additional recoils do not change the interference fringes, we have amplified the effective recoil shift without altering the precision with which we can determine the resonance centers, thereby improving the final measurement resolution linearly proportional to N.

1.2 Overview of this thesis

In Chapter 2, I derive general expressions for the phase of the interferometers and describe the different interferometer geometries. I also discuss our particular technique using adiabatic dark-state evolution to transfer atoms between the two hyperfine ground states and its implications to interferometry. Chapter 3 covers the details of the experimental apparatus, focusing mostly on the generation of the crucial Raman lasers which impart the recoils and build the interferometers. Since this is the third thesis from this experiment, Chapter 4 discusses the more significant changes in the apparatus and overall improvements. In Chapter 5, I briefly discuss what the data look like and how we acquire and process them. Most important for a precision measurement is the discussion of the tests for systematic errors, which I plod through one by one in Chapter 6. Finally, Chapter 7 concludes with a presentation of the final value, and Chapter 8 presents a brief discussion of possible future improvements.

Chapter 2

Theory

2.1 Atom interferometry

2.1.1 Interferometer Phase

To calculate the phase of an interferometer from initial splitting to final recombining we consider the interaction of a single atom of mass m with a laser field of wavevector $\mathbf{k}_\mathrm{L}$ and frequency ω_L, whose wavefront propagation can be described by $\exp\left[i(\mathbf{k}_\mathrm{L} \cdot \mathbf{r} - \omega_\mathrm{L} t - \phi_\mathrm{L})\right]$. Since our laser fields are well-collimated and highly directional, we follow only one spatial dimension, the beam direction, and assume $\mathbf{k}_\mathrm{L} = k_\mathrm{L}\hat{\mathbf{z}}$, in the vertical direction. We assume that the atom has well-defined initial momentum and two internal states, $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$, with energies $\hbar\omega_\mathrm{a}$ and $\hbar\omega_\mathrm{b}$, respectively.

When the laser light is off, we assume that the phase evolution of the atomic wavefunction can be described by $S_\mathrm{cl}/\hbar$ where $S_\mathrm{cl} = \int \mathrm{d}t\, L(z, \dot{z})$ is the classical action. For an atom in a uniform gravitational field with internal energy levels ω_i, the Lagrangian

$$L(z, \dot{z}) = \frac{1}{2} m \dot{z}^2 - mgz - \hbar\omega_i \tag{2.1}$$

gives the classical solutions for velocity $v(t) = \dot{z}(t)$ and position $z(t)$. Evaluating the action along the path described by this classical solution gives the classical action which depends only on the endpoints z_a and z_b [12, 13].

$$\begin{aligned} S_{\mathrm{cl}}(z_{\mathrm{b}}t_{\mathrm{b}}, z_{\mathrm{a}}t_{\mathrm{a}}) &= \int_{t_{\mathrm{a}}}^{t_{\mathrm{b}}} \mathrm{d}t \left[\tfrac{1}{2}mv(t)^2 - mgz(t) - \hbar\omega_i\right] \\ &= \frac{m}{2}\frac{(z_{\mathrm{b}}-z_{\mathrm{a}})^2}{t_{\mathrm{b}}-t_{\mathrm{a}}} - \frac{mg}{2}(z_{\mathrm{b}}+z_{\mathrm{a}})(t_{\mathrm{b}}-t_{\mathrm{a}}) \\ &\quad -\frac{mg^2}{24}(t_{\mathrm{b}}-t_{\mathrm{a}})^3 - \hbar\omega_i(t_{\mathrm{b}}-t_{\mathrm{a}}) \end{aligned} \quad (2.2)$$

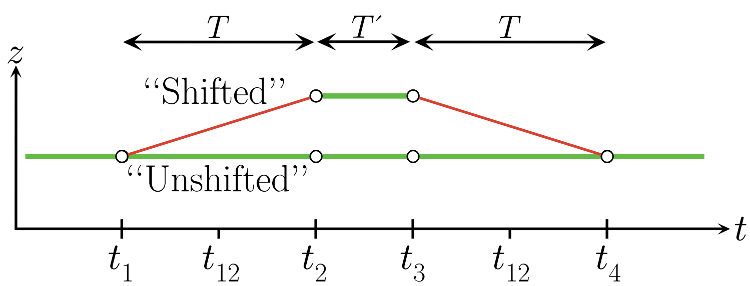


Figure 2.1: **Interferometer geometry [1].** An atom at rest and initially in state $|\mathrm{a}\rangle$ is transfered to a superposition of states $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$, represented with thick and thin lines, respectively. Due to momentum conservation, these internal states are coupled to different momentum states. Plotting position versus time and neglecting gravity, these parts of the atomic wavefunction separate in time into two different paths. The path that deviates from the input trajectory will be called the "shifted" path, while the other path will be referred to as the "unshifted" path.

For all interferometer geometries (enumerated [1], [2], [3], and [4]), we assume that the atom starts in the internal state $|\mathrm{a}\rangle$ with velocity v_1 in the z-direction. The first $\pi/2$-pulse puts the atom in a superposition of states $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$. In order to conserve momentum, the velocity of the part of the atomic wavefunction in state $|\mathrm{b}\rangle$ must differ from the velocity of the part of the atomic wavefunction in state $|\mathrm{a}\rangle$ by $\mathbf{v}_{\mathrm{r}} = \hbar\mathbf{k}_{\mathrm{L}}/m = \pm(\hbar k_{\mathrm{L}}/m)\hat{\mathbf{z}}$, where the sign is determined by the direction of the laser wavevector, parallel or anti-parallel to $\hat{\mathbf{z}}$. Because of this velocity difference, as depicted in Figure 2.1 these two parts of the wavefunction separate spatially into two paths. We label the $|\mathrm{b}\rangle$-state path "S" for "shifted" and the $|\mathrm{a}\rangle$-state path, which does not change velocity during the first $\pi/2$-pulse, "U" for "unshifted".

For the unshifted and shifted paths with the four $\pi/2$-pulses occuring at times t_i

we calculate the atom's velocity v_i and position z_i just *before* the i-th $\pi/2$-pulse.

$$\begin{array}{l|l} v_{2\mathrm{U}} = v_{1\mathrm{U}} - g\,(t_2 - t_1) & v_{2\mathrm{S}} = v_{1\mathrm{S}} - g\,(t_2 - t_1) + v_\mathrm{r} \\ v_{3\mathrm{U}} = v_{1\mathrm{U}} - g\,(t_3 - t_1) & v_{3\mathrm{S}} = v_{1\mathrm{S}} - g\,(t_3 - t_1) \\ v_{4\mathrm{U}} = v_{1\mathrm{U}} - g\,(t_4 - t_1) & v_{4\mathrm{S}} = v_{1\mathrm{S}} - g\,(t_4 - t_1) - v_\mathrm{r} \end{array} \tag{2.3}$$

$$\begin{array}{l|} z_{2\mathrm{U}} = z_{1\mathrm{U}} + v_{1\mathrm{U}}(t_2 - t_1) - \frac{1}{2}gt^2\Big|_0^{t_2-t_1} \\ z_{3\mathrm{U}} = z_{2\mathrm{U}} + v_{1\mathrm{U}}(t_3 - t_2) - \frac{1}{2}gt^2\Big|_{t_2-t_1}^{t_3-t_1} \\ z_{4\mathrm{U}} = z_{3\mathrm{U}} + v_{1\mathrm{U}}(t_4 - t_3) - \frac{1}{2}gt^2\Big|_{t_3-t_1}^{t_4-t_1} \end{array}$$
$$\begin{array}{|l} z_{2\mathrm{S}} = z_{1\mathrm{S}} + (v_{1\mathrm{S}} + v_\mathrm{r})(t_2 - t_1) - \frac{1}{2}gt^2\Big|_0^{t_2-t_1} \\ z_{3\mathrm{S}} = z_{2\mathrm{S}} + (v_{1\mathrm{S}})(t_3 - t_2) - \frac{1}{2}gt^2\Big|_{t_2-t_1}^{t_3-t_1} \\ z_{4\mathrm{S}} = z_{3\mathrm{S}} + (v_{1\mathrm{S}} - v_\mathrm{r})(t_4 - t_3) - \frac{1}{2}gt^2\Big|_{t_3-t_1}^{t_4-t_1} \end{array} \tag{2.4}$$

We now assert that $t_4 - t_3 = t_2 - t_1 = T$, which is required for the two paths to intersect at the fourth $\pi/2$-pulse[1]. We also define the interval $t_3 - t_2 = T'$ which will eventually contain N π-pulses but for now just represents some arbitrary delay between the first and second halves of the interferometer. Note that $v_{1\mathrm{S}} = v_{1\mathrm{U}} = v_1$. The velocity and position thus become

$$\begin{array}{l|l} v_{2\mathrm{U}} = v_1 - g\,T & v_{2\mathrm{S}} = v_1 - g\,T + v_\mathrm{r} \\ v_{3\mathrm{U}} = v_1 - g\,(T + T') & v_{3\mathrm{S}} = v_1 - g\,(T + T') \\ v_{4\mathrm{U}} = v_1 - g\,(2T + T') & v_{4\mathrm{S}} = v_1 - g\,(2T + T') - v_\mathrm{r} \end{array} \tag{2.5}$$

[1]In principle, if the two interferometer paths do not intersect in position (and velocity) space, they will not interfere. However, as long as the spread of the individual atomic wavefunctions is larger than the gap between the interferometer paths at the final interference point, there will still be interference. The two paths can be thought of as originating from two different points within the atomic wavepacket such that the two paths overlap at the final interference point. The possible phase errors from this effect are discussed in Section 6.6.3.

$$
\left.\begin{aligned}
z_{2\mathrm{U}} &= z_{1\mathrm{U}} + v_1 T - \tfrac{1}{2} g T^2 \\
z_{3\mathrm{U}} &= z_{2\mathrm{U}} + v_1 T' - \tfrac{1}{2} g \Big[(T+T')^2 - T^2\Big] \\
&= z_{2\mathrm{U}} + v_1 T' - \tfrac{1}{2} g \Big[2TT' + T'^2\Big] \\
z_{4\mathrm{U}} &= z_{3\mathrm{U}} + v_1 T - \tfrac{1}{2} g \Big[(2T+T')^2 - (T+T')^2\Big] \\
&= z_{3\mathrm{U}} + v_1 T - \tfrac{1}{2} g \Big[3T^2 + 2TT'\Big]
\end{aligned}\right|
\qquad
\left|\begin{aligned}
z_{2\mathrm{S}} &= z_{1\mathrm{S}} + (v_1 + v_\mathrm{r}) T - \tfrac{1}{2} g T^2 \\
z_{3\mathrm{S}} &= z_{2\mathrm{S}} + v_1 T' - \tfrac{1}{2} g \Big[2TT' + T'^2\Big] \\
z_{4\mathrm{S}} &= z_{3\mathrm{S}} + (v_1 - v_\mathrm{r}) T - \tfrac{1}{2} g \Big[3T^2 + 2TT'\Big]
\end{aligned}\right.
\tag{2.6}
$$

Evaluating the action over the paths described in equation (2.6) we can calculate the relative phase shift of the atomic wavefunction between the two interferometer paths. Between the first and second $\pi/2$-pulses, from equation (2.2) we have

$$
\begin{aligned}
\frac{1}{\hbar}\left[S_\mathrm{cl}(\mathrm{shifted12}) - S_\mathrm{cl}(\mathrm{unshifted12})\right] &= \left[S_\mathrm{cl}(\mathrm{S12}) - S_\mathrm{cl}(\mathrm{U12})\right]/\hbar \;=\; \Delta S_\mathrm{cl}(12)/\hbar \\
= \frac{m}{2\hbar}\Bigg\{ \frac{(z_{2\mathrm{S}} - z_{1\mathrm{S}})^2}{(t_2 - t_1)} &- \frac{(z_{2\mathrm{U}} - z_{1\mathrm{U}})^2}{(t_2 - t_1)} - g\,(t_2 - t_1)\Big[(z_{2\mathrm{S}} + z_{1\mathrm{S}}) - (z_{2\mathrm{U}} + z_{1\mathrm{U}})\Big]\Bigg\} \\
&\qquad - \Big[\omega_\mathrm{b}(t_2 - t_1) - \omega_\mathrm{a}(t_2 - t_1)\Big] \\
= \frac{m}{2\hbar}\left\{ \frac{\Big((v_1 + v_\mathrm{r})T - \frac{1}{2} g\,T^2\Big)^2}{T} \right. &\left. - \frac{\Big(v_1 T - \frac{1}{2} g\,T^2\Big)^2}{T} - g\,T\,[v_\mathrm{r} T] \right\} - (\omega_\mathrm{b} - \omega_\mathrm{a}) T \\
= \frac{m}{2\hbar}\Big\{2 v_1 v_\mathrm{r} T + v_\mathrm{r}^2 T &- v_\mathrm{r} g\,T^2 - v_\mathrm{r} g\,T^2\Big\} - \omega_\mathrm{ab} T \\
= \frac{m}{2\hbar} v_\mathrm{r} T \left\{2 v_1 + v_\mathrm{r} - 2 g\,T\right\} &- \omega_\mathrm{ab} T
\end{aligned}
\tag{2.7}
$$

where $\omega_\mathrm{ab} = \omega_\mathrm{b} - \omega_\mathrm{a}$ is the frequency difference between the two internal energy levels. Note that because the third term in equation (2.2) is proportional only to the time separation between the two pulses and not the position, it cancels immediately between the two interferometer paths. Between the second and third $\pi/2$-pulses, we have

$$
\left[S_\mathrm{cl}(\mathrm{S23}) - S_\mathrm{cl}(\mathrm{U23})\right]/\hbar \;=\; \Delta S_\mathrm{cl}(23)/\hbar
$$

$$
\begin{aligned}
&= \frac{m}{2\hbar}\left\{ \frac{(z_{3S}-z_{2S})^2}{(t_3-t_2)} - \frac{(z_{3U}-z_{2U})^2}{(t_3-t_2)} - g\,(t_3-t_2)\Big[(z_{3S}+z_{2S})-(z_{3U}+z_{2U})\Big]\right\} \\
&\qquad - \Big[\omega_a(t_3-t_2)-\omega_a(t_3-t_2)\Big] \\
&= \frac{m}{2\hbar}\left\{0 - g\,T'\,[2v_r T]\right\} - 0 \\
&= \frac{m}{2\hbar} v_r T\left\{-2g\,T'\right\} \qquad (2.8)
\end{aligned}
$$

And finally, the phase difference between the shifted and unshifted paths for the last segment is

$$
\begin{aligned}
&[S_{cl}(S34) - S_{cl}(U34)]/\hbar \;=\; \Delta S_{cl}(34)/\hbar \\
&= \frac{m}{2\hbar}\left\{ \frac{(z_{4S}-z_{3S})^2}{(t_4-t_3)} - \frac{(z_{4U}-z_{3U})^2}{(t_4-t_3)} - g\,(t_4-t_3)\Big[(z_{4S}+z_{3S})-(z_{4U}+z_{3U})\Big]\right\} \\
&\qquad - \Big[\omega_b(t_4-t_3)-\omega_a(t_4-t_3)\Big] \\
&= \frac{m}{2\hbar}\left\{ \frac{\Big((v_1-v_r)T - \frac{1}{2}g\,T(3T+2T')\Big)^2}{T} - \frac{\Big(v_1 T - \frac{1}{2}g\,T(3T+2T')\Big)^2}{T} - g\,T\,[v_r T]\right\} \\
&\qquad - \Big[(\omega_b-\omega_a)T\Big] \\
&= \frac{m}{2\hbar}\Big\{-2v_1 v_r T + v_r^2 T + v_r g\,T(3T+2T') - v_r g\,T^2\Big\} - \omega_{ab}T \\
&= \frac{m}{2\hbar} v_r T\left\{-2v_1 + v_r + 2g(T+T')\right\} - \omega_{ab}T \qquad (2.9)
\end{aligned}
$$

Summing equations (2.7) through (2.9) we have the overall phase difference between the shifted (Φ_S) and unshifted (Φ_U) interferometer paths due to the evolution of the atomic wavefunction

$$
\begin{aligned}
\Phi_{\boxed{1}}(\text{atom}) &= \Phi_S - \Phi_U \;=\; [\Delta S_{cl}(12) + \Delta S_{cl}(23) + \Delta S_{cl}(34)]/\hbar \\
&= \frac{m}{2\hbar} v_r T\{2v_r\} - 2\omega_{ab}T \\
&= \frac{m v_r^2 T}{\hbar} - 2\omega_{ab}T \qquad (2.10)
\end{aligned}
$$

As contrasted with other more symmetric interferometer geometries [14, 15], this result for our interferometers is non-zero. Note that because we have evaluated the *difference* between the two interferometer paths, the final expression in equation (2.10) is independent of the initial velocity v_1 and the gravitational acceleration g.

In addition to the phase evolution between the pulses, to calculate the complete phase expression for an interferometer geometry, we must include also the effect of the light at each pulse for each path of each interferometer. As shown in Table 2.1, we assume that whenever the atom changes state, that part of the atomic wavefunction acquires a phase identical to the optical phase at that point in space and time. Note that this model does not refer to a particular technique used to transfer atoms between atomic states, such as adiabatic passage or off-resonant Raman transfer.

Table 2.1: **Change in the atomic wavefunction due to the interaction with the laser field.** U_{ij} are real transition amplitudes that for our purposes can be assumed to be unity. Note that only when the internal state of the atom changes does the light field imprint its phase on the atomic wavefunction.

Initial State	Final State	Momentum Change	Multiplying factor
a	a	0	U_{aa}
a	b	$+\hbar k_{\mathrm{L}}$	$U_{\mathrm{ab}} \exp\left[-i(k_{\mathrm{L}}z - \omega_{\mathrm{L}}t - \phi_{\mathrm{L}})\right]$
b	a	$-\hbar k_{\mathrm{L}}$	$U_{\mathrm{ba}} \exp\left[+i(k_{\mathrm{L}}z - \omega_{\mathrm{L}}t - \phi_{\mathrm{L}})\right]$
b	b	0	U_{bb}

For interferometer geometry $\boxed{1}$, we first calculate the effect of the four $\pi/2$-pulses by applying the rules given in Table 2.1 to each of the vertices of the shifted and unshifted interferometer paths. To apply these rules, we must know the phase of the optical wavefront at each vertex. To better conceptualize the contributions of the individual terms, we separate the optical phase into three parts: the "kz" term, the "$-\omega t$" term, and the "ϕ" term and evaluate the complete interferometer for each term separately.

Movement along the optical wavefronts: the kz term

As an atom moves in space along a laser beam that is fixed in space with respect to some absolute reference, the atom experiences an optical phase that varies proportional to $k_{\mathrm{L}}z$, where k_{L} is the magnitude of the laser's wavevector and z is some position along that vector. For now we assume that the momentum carried by the laser beams for each $\pi/2$-pulse differs only in direction and not in magnitude: $|k_1| = |k_2| = |k_3| = |k_4| = k_{\mathrm{L}} = k$. For interferometer geometry $\boxed{1}$, during the first two $\pi/2$-pulses the laser field propagates upward: $k_1 = k_2 = +k$, and during the second two $\pi/2$-pulses switches direction: $k_3 = k_4 = -k$. Using Table 2.1 to evaluate the kz term vertex by vertex for both paths of interferometer $\boxed{1}$ we have

Geometry $\boxed{1}$	$\pi/2$-pulse #1	#2	#3	#4
Direction of $\lvert\mathbf{k}\rvert$	Up	Up	Down	Down
Unshifted path	0	0	0	0
Shifted path	$+k_1 z_{1\mathrm{S}}$ $= +kz_{1\mathrm{S}}$	$-k_2 z_{2\mathrm{S}}$ $= -kz_{2\mathrm{S}}$	$+k_3 z_{3\mathrm{S}}$ $= -kz_{3\mathrm{S}}$	$-k_4 z_{4\mathrm{S}}$ $= +kz_{4\mathrm{S}}$

At each vertex on the unshifted path, because the atom does not change state, no phase is imprinted on the atomic wavefunction. For the shifted path, however, each vertex contributes; the first and the third add because the transition is $|\mathrm{a}\rangle \to |\mathrm{b}\rangle$, while the second and fourth subtract because the transition is $|\mathrm{b}\rangle \to |\mathrm{a}\rangle$.

Using the position results from equation (2.6). The difference between the shifted and unshifted paths for kz term is

$$\begin{aligned}
\Phi_{\mathrm{S}} - \Phi_{\mathrm{U}} &= k\left[-(z_{2\mathrm{S}} - z_{1\mathrm{S}}) + (z_{4\mathrm{S}} - z_{3\mathrm{S}})\right] \\
&= k\left[-\left((v_1 + v_{\mathrm{r}})T - \tfrac{1}{2}gT^2\right) + \left((v_1 - v_{\mathrm{r}})T - \tfrac{1}{2}g\,(3T^2 + 2TT')\right)\right] \\
&= k\Big[-2v_{\mathrm{r}} - g\,(T + T')\Big]T \qquad (2.11)
\end{aligned}$$

which is proportional to the recoil velocity v_{r} and the local acceleration g from gravity.

The time evolution of the optical phase: the $-\omega t$ term

In all precision interferometry experiments, the time evolution of the system being measured is metered relative to the evolution of a stable reference oscillator, which in our case with cesium atoms is the light field. After the first $\pi/2$-pulse when the light is off, the relative phase between the two internal pure atomic states $|\mathrm{a}\rangle$ and

$|b\rangle$ and their respective velocities evolves according to their energy difference. This energy difference includes the ground state hyperfine splitting of cesium $\omega_{\rm ab}$ and the kinetic energy difference due to a velocity change $v_{\rm r}$ from the absorption or emission of a photon. The $-\omega t$ term acts as the reference oscillator that is compared to this internal phase evolution of the atomic wavefunction. As discussed in Section 1, due to the two-photon process, the resonance condition is defined by the difference between the absolute frequencies of the two fields, so the laser frequency $\omega_{\rm L}$ in the optical phase expression represents here the difference between the two laser frequencies. This frequency difference is set to cancel both the hyperfine splitting, which is defined, and the recoil energy, which we are trying to measure. If the value $\bar{\omega}_{\rm rec} = \bar{k}\bar{v}_{\rm r}$ for the recoil frequency used to set the oscillator frequency differs from the true value $\omega_{\rm rec}$, the atomic wavefunction and the reference oscillator will evolve at slightly different rates and produce a phase shift after some time interval. It is this phase shift that we measure at the end of the interferometers which tells us how much our current value $\bar{\omega}_{\rm rec}$ for the recoil velocity differs from the true value $\omega_{\rm rec}$.

To track the phase evolution of the reference oscillator, we must know its frequency at all times. From the atomic velocity given in equation (2.5) we can evaluate the resonance condition in equation (1.15) for each $\pi/2$-pulse and calculate the frequencies of the laser fields. During the first $\pi/2$-pulse, because the atom is in state $|a\rangle$ moving with velocity v_1, the laser fields are two-photon resonant when their frequency difference is $\omega = \omega_{\rm ab} + kv_1 + \frac{1}{2}kv_{\rm r}$. The last term compensates for the change in kinetic energy due to the stimulated absorption and emission of two photons from and into the laser fields. For the second $\pi/2$-pulse, the beam direction is the same but the atom is in a superposition state with velocity $v_{\rm 2U}$ for the $|a\rangle$ part and $v_{\rm 2S} = v_{\rm 2U} + v_{\rm r}$ for the $|b\rangle$ part. The resonance condition for the part of the atom in the $|b\rangle$ state is $\omega = \omega_{\rm ab} + k(v_{\rm 2U} + v_{\rm r}) - \frac{1}{2}kv_{\rm r} = \omega_{\rm ab} + kv_{\rm 2U} + \frac{1}{2}kv_{\rm r}$, identical to the resonance condition for the $|a\rangle$ state. Because the laser field does not change direction between the first and second $\pi/2$-pulses, the laser field plus the atom still make up the same closed system. Thus, except for the effect of gravity, the atom still has the exact same resonance condition. For interferometer geometry $\boxed{1}$, the resonance frequency ω_i for the ith $\pi/2$-pulse is

$$\begin{aligned}
\omega_1 - \omega_{\rm ab} &= \delta + \bar{k_1}\bar{v_1} + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} = \delta + \bar{k}\bar{v_1} + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} & (2.12)\\
\omega_2 - \omega_{\rm ab} &= \delta + \bar{k_2}\bar{v_2} + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} = \delta + \bar{k}\left[\bar{v_1} - \bar{g}\,T\right] + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} & (2.13)\\
\omega_3 - \omega_{\rm ab} &= \delta + \bar{k_3}\bar{v_3} + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} = \delta - \bar{k}\left[\bar{v_1} - \bar{g}\left(T + T'\right)\right] + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} & (2.14)\\
\omega_4 - \omega_{\rm ab} &= \delta + \bar{k_4}\bar{v_4} + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} = \delta - \bar{k}\left[\bar{v_1} - \bar{g}\left(2T + T'\right)\right] + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} & (2.15)
\end{aligned}$$

where δ represents any frequency offset from resonance, intentional or otherwise. Here the bars indicate our best guess for each quantity, as opposed to their actual value.

Let $\phi_i(\omega)$ be the phase of the $-\omega t$ term at the time of the ith $\pi/2$-pulse, then the total phase expression of interferometer $\boxed{1}$ for the time varying part of the optical phase is

$$\begin{aligned}
\Phi_{\rm S} - \Phi_{\rm U} &= +\phi_1(\omega) - \phi_2(\omega) + \phi_3(\omega) - \phi_4(\omega)\\
&= \phi_1(\omega) - \left[\phi_1(\omega) + \Delta\phi_{12}(\omega)\right] + \left[\phi_1(\omega) + \Delta\phi_{12}(\omega) + \Delta\phi_{23}(\omega)\right]\\
&\qquad - \left[\phi_1(\omega) + \Delta\phi_{12}(\omega) + \Delta\phi_{23}(\omega) + \Delta\phi_{34}(\omega)\right]\\
&= -\left[\Delta\phi_{12}(\omega)\right] - \left[\Delta\phi_{34}(\omega)\right] \qquad (2.16)
\end{aligned}$$

where $\Delta\phi_{ij}(\omega)$ is the change in time-dependent phase between the ith and jth $\pi/2$-pulses. We have applied the rules in Table 2.1: the first and third $\pi/2$-pulses add, the second and fourth $\pi/2$-pulses subtract, and the unshifted path does not contribute. Note that because of the geometry of our four $\pi/2$-pulse interferometers, the phase evolution of this $-\omega t$ term contributes only between the first and second and between the third and fourth $\pi/2$-pulses when the atom is in a superposition state.

To evaluate this expression, we must introduce our experimental method for changing from one frequency to the next. As a step-wise approximation to changing the frequency continuously to compensate for gravity, we change frequencies at the midpoints, t_{12} and t_{34}, of the $\pi/2$-pulse pairs (see Figure 2.1). Thus, equation (2.16) becomes

$$-\Big[-\omega_1\left(t_{12} - t_1\right) - \omega_2\left(t_2 - t_{12}\right)\Big] - \Big[-\omega_3\left(t_{34} - t_3\right) - \omega_4\left(t_4 - t_{34}\right)\Big]$$

Inserting the frequencies from equations (2.12) through (2.15) into this equation we have

$$
\begin{aligned}
\Phi_{\mathrm{S}} - \Phi_{\mathrm{U}} &= \left[\delta + \omega_{\mathrm{ab}} + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}}\right](t_{12} - t_1) \\
&\quad + \left[\delta + \omega_{\mathrm{ab}} + \bar{k}\bar{v}_1 - \bar{k}\bar{g}\,T + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}}\right](t_2 - t_{12}) \\
&\quad + \left[\delta + \omega_{\mathrm{ab}} - \bar{k}\bar{v}_1 + \bar{k}\bar{g}\,(T + T') + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}}\right](t_{34} - t_3) \\
&\quad + \left[\delta + \omega_{\mathrm{ab}} - \bar{k}\bar{v}_1 + \bar{k}\bar{g}\,(2T + T') + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}}\right](t_4 - t_{34})
\end{aligned}
\tag{2.17}
$$

With the same constraints on the timing of the $\pi/2$-pulses discussed in the previous section (namely, $t_4 - t_3 = t_2 - t_1 = T$ and $t_3 - t_2 = T'$), we have finally

$$
\begin{aligned}
\Phi_{\mathrm{S}} - \Phi_{\mathrm{U}} &= \left[\delta + \omega_{\mathrm{ab}} + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}}\right]T - \bar{k}\bar{g}\,T\,(t_2 - t_{12}) \\
&\quad + \left[\delta + \omega_{\mathrm{ab}} - \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_{\mathrm{r}} + \bar{k}\bar{g}\,(T + T')\right]T + \bar{k}\bar{g}\,T\,(t_4 - t_{34}) \\
&= \left[2(\omega_{\mathrm{ab}} + \delta) + \bar{k}\bar{v}_{\mathrm{r}} + \bar{k}\bar{g}\,(T + T')\right]T + \bar{k}\bar{g}\,T\,[(t_4 - t_2) - (t_{34} - t_{12})] \\
&= \left[2(\omega_{\mathrm{ab}} + \delta) + \bar{k}\bar{v}_{\mathrm{r}} + \bar{k}\bar{g}\,(T + T')\right]T + \bar{k}\bar{g}\,T\,[(T + T') - (T + T')] \\
&= \left[2(\omega_{\mathrm{ab}} + \delta) + \bar{k}\bar{v}_{\mathrm{r}} + \bar{k}\bar{g}\,(T + T')\right]T
\end{aligned}
\tag{2.18}
$$

Additional optical phase: the ϕ term

Beyond the fundamental phase change due to the time evolution of the optical wavefronts and the motion of the atom along the laser field, any additional phase shift will also contribute to the final phase result. Sources for such a phase shift might include imperfections in the transfer, shifts of the atomic levels due to the presence of the light fields during the pulses, or any of the many noise sources which cause the optical phase to deviate from precisely $kz - \omega t$. Independent of its origin, however, any additional phase shift during the light pulses will contribute as

$$\Phi_{\mathrm{S}} - \Phi_{\mathrm{U}} = \phi_1 - \phi_2 + \phi_3 - \phi_4 \tag{2.19}$$

Combining equations (2.11) and (2.18), we can now construct the final interferometer phase Φ(light) due to the light fields interacting with the atoms during the four $\pi/2$-pulses. For now, we leave out any additional phase shifts referred to in equation (2.19).

$$\begin{aligned}
&\Phi_{\boxed{1}}(\text{light}) \\
&= k\left[-2v_\text{r} - g\,(T+T')\right]T + \left[2(\omega_\text{ab}+\delta) + \bar{k}\bar{v}_\text{r} + \bar{k}\bar{g}\,(T+T')\right]T \\
&= \left[-kv_\text{r} - (kv_\text{r} - \bar{k}\bar{v}_\text{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2(\omega_\text{ab}+\delta)T
\end{aligned} \tag{2.20}$$

Combining this result with the total phase shift from the free evolution of the atom between the pulses in equation (2.10), we have the complete phase difference for interferometer geometry $\boxed{1}$.

$$\begin{aligned}
\Phi_{\boxed{1}} &= \Phi_{\boxed{1}}(\text{atom}) + \Phi_{\boxed{1}}(\text{light}) \\
&= \frac{mv_\text{r}T}{\hbar}v_\text{r} - 2\omega_\text{ab}T \\
&\qquad + \left[-kv_\text{r} - (kv_\text{r} - \bar{k}\bar{v}_\text{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2(\omega_\text{ab}+\delta)T \\
&= \frac{mv_\text{r}T}{\hbar}\frac{\hbar k}{m} + \left[-kv_\text{r} - (kv_\text{r} - \bar{k}\bar{v}_\text{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2\delta T \\
&= \left[-(kv_\text{r} - \bar{k}\bar{v}_\text{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2\delta T
\end{aligned} \tag{2.21}$$

If our guesses for the values of the recoil velocity $\bar{v}_\text{r}$ and the acceleration due to gravity $\bar{g}$ are correct and $\delta = 0$, the net phase shift will be zero and the atom will emerge in state $|\text{a}\rangle$ with unity probability. Any difference between our guesses and the actual values will reduce this probability and be observed as a shift in the interferometer fringe. Note that in addition to the recoil velocity, this interferometer geometry is also sensitive to the local gravitational acceleration. To make the overall measurement independent of small inaccuracies in the value for g, we make a second measurement with a slightly modified geometry.

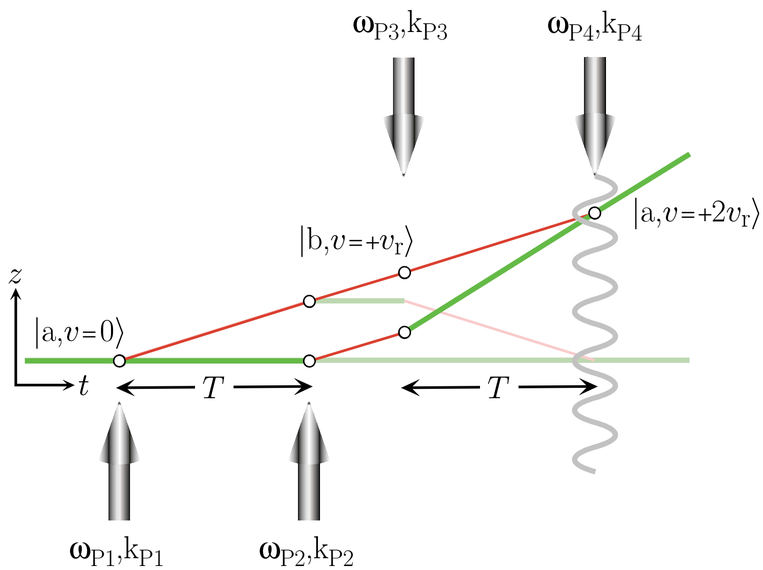


Figure 2.2: **The fundamental recoil measurement** is a difference between two interferometers. By selecting the other internal state (in this case |b⟩) after the second $\pi/2$-pulse, we can generate another interferometer, conjugate to the first, shown with light lines. Although the final interferometer phase also depends on the laser frequencies during the pulses and the atom's free evolution during time T, the recoil measurement can be understood as counting the number of laser wavefronts, shown for the last $\pi/2$-pulse, between the final interference points of the two interferometers.

2.1.2 Conjugate Interferometer

In order to make the overall measurement sensitive only to deviations in the recoil shift, we can measure the phase from a slightly modified interferometer geometry. As we will show below, interferometer geometry $\boxed{2}$ (see Figure 2.2) has the same sensitivity as geometry $\boxed{1}$ except the recoil terms enter with the opposite sign. By taking the difference between the results from geometries $\boxed{1}$ and $\boxed{2}$, we remove the sensitivity to the local gravitational acceleration and to any frequency detuning from the two-photon resonance condition. Interferometers $\boxed{1}$ and $\boxed{2}$ are thus "conjugate" interferometer geometries; *both* must be measured in order to arrive at a value for the recoil shift independent of g.

As depicted in Figure 2.2, geometry $\boxed{2}$ is identical to its conjugate geometry $\boxed{1}$, except that after the first two $\pi/2$-pulses, the atom is left in the other hyperfine state: state $|b\rangle$ instead $|a\rangle$. Geometries $\boxed{3}$ and $\boxed{4}$ (see Figure 2.5) are constructed from $\boxed{1}$ and $\boxed{2}$ by reversing the direction of all of the laser beams. We will refer to the conjugate interferometer pair $\boxed{1}$ and $\boxed{2}$ as the "normal" interferometers and the conjugate pair $\boxed{3}$ and $\boxed{4}$ as the "inverted" interferometers.

To calculate the final phase for interferometer geometry $\boxed{2}$, we follow the same procedure as in the previous section. Equations (2.5) and (2.6) for the velocity and position become

$$\begin{array}{l|l} v_{2\mathrm{U}} = v_1 - g\,T & v_{2\mathrm{S}} = v_1 - g\,T + v_{\mathrm{r}} \\ v_{3\mathrm{U}} = v_1 - g\,(T+T') + v_{\mathrm{r}} & v_{3\mathrm{S}} = v_1 - g\,(T+T') + v_{\mathrm{r}} \\ v_{4\mathrm{U}} = v_1 - g\,(2T+T') + 2v_{\mathrm{r}} & v_{4\mathrm{S}} = v_1 - g\,(2T+T') + v_{\mathrm{r}} \end{array} \tag{2.22}$$

$$\begin{aligned} z_{2\mathrm{U}} &= z_{1\mathrm{U}} + v_1 T - \tfrac{1}{2} g T^2 \\ z_{3\mathrm{U}} &= z_{2\mathrm{U}} + (v_1 + v_{\mathrm{r}})T' - \tfrac{1}{2} g \left[2TT' + T'^2\right] \\ z_{4\mathrm{U}} &= z_{3\mathrm{U}} + (v_1 + 2v_{\mathrm{r}})T - \tfrac{1}{2} g \left[3T^2 + 2TT'\right] \\ z_{2\mathrm{S}} &= z_{1\mathrm{S}} + (v_1 + v_{\mathrm{r}})T - \tfrac{1}{2} g T^2 \\ z_{3\mathrm{S}} &= z_{2\mathrm{S}} + (v_1 + v_{\mathrm{r}})T' - \tfrac{1}{2} g \left[2TT' + T'^2\right] \\ z_{4\mathrm{S}} &= z_{3\mathrm{S}} + (v_1 + v_{\mathrm{r}})T - \tfrac{1}{2} g \left[3T^2 + 2TT'\right] \end{aligned} \tag{2.23}$$

From this trajectory, we calculate the difference between the action along the shifted and unshifted paths. $\Delta S_{\mathrm{cl}}(12)$ and $\Delta S_{\mathrm{cl}}(23)$ are the same, but $\Delta S_{\mathrm{cl}}(34)$ from equation (2.9) becomes

$$\begin{aligned} &\Delta S_{\mathrm{cl}}(34)/\hbar \\ &= \frac{m}{2\hbar}\left\{ \frac{\left((v_1 + v_{\mathrm{r}})T - \frac{1}{2} g\,T(3T+2T')\right)^2}{T} - \frac{\left((v_1 + 2v_{\mathrm{r}})T - \frac{1}{2} g\,T(3T+2T')\right)^2}{T} \right. \\ &\qquad \left. - g\,T\,[v_{\mathrm{r}} T] \right\} - \left[(\omega_{\mathrm{b}} - \omega_{\mathrm{a}})T\right] \\ &= \frac{m}{2\hbar}\left\{-2v_1 v_{\mathrm{r}} T + v_{\mathrm{r}}^2 T - 3v_{\mathrm{r}} g\,T(3T+2T') - v_{\mathrm{r}} g\,T^2\right\} - \omega_{\mathrm{ab}} T \\ &= \frac{m}{2\hbar} v_{\mathrm{r}} T\left\{-2v_1 - 3v_{\mathrm{r}} + 2g(T+T')\right\} - \omega_{\mathrm{ab}} T \end{aligned} \tag{2.24}$$

and the overall phase difference between the shifted and unshifted paths due to the evolution of the atomic wavefunction is

$$\begin{aligned}
\Phi_{\boxed{2}}(\text{atom}) &= \Phi_\text{S} - \Phi_\text{U} = \left[\Delta S_\text{cl}(12) + \Delta S_\text{cl}(23) + \Delta S_\text{cl}(34)\right]/\hbar \\
&= \frac{m}{2\hbar} v_\text{r} T\{-2v_\text{r}\} - 2\omega_\text{ab} T \\
&= -\frac{m v_\text{r}^2 T}{\hbar} - 2\omega_\text{ab} T \qquad (2.25)
\end{aligned}$$

differing from the same expression for interferometer $\boxed{1}$ in equation (2.10) only in the sign of the first term.

To calculate the effect of the light pulses, we again evaluate the kz term, the $-\omega t$ term, and the ϕ term separately. As before, we assume $|k_1| = |k_2| = |k_3| = |k_4| = k_\text{L} = k$, with the same beam directions as with geometry $\boxed{1}$. At each $\pi/2$-pulse for geometry $\boxed{2}$, we have

Geometry $\boxed{2}$	$\pi/2$-pulse #1	#2	#3	#4
Direction of $\lvert\mathbf{k}\rvert$	Up	Up	Down	Down
Unshifted path	0	$+k_2 z_{2\text{S}}$ $= +kz_{2\text{S}}$	$-k_3 z_{3\text{S}}$ $= +kz_{3\text{S}}$	0
Shifted path	$+k_1 z_{1\text{S}}$ $= +kz_{1\text{S}}$	0	0	$-k_4 z_{4\text{S}}$ $= +kz_{4\text{S}}$

Unlike geometry $\boxed{1}$, both paths for this geometry contribute to the final expression. Along the unshifted path, the atom changes state during the second ($|\text{a}\rangle$ to $|\text{b}\rangle$) and third ($|\text{b}\rangle$ to $|\text{a}\rangle$) $\pi/2$-pulses, adding during the second and subtracting during the third. Along the shifted path, only the first and fourth $\pi/2$-pulse contribute, the first adding and the fourth subtracting. Using the position results from equation (2.23). The difference between the shifted and unshifted paths for kz term is

$$\begin{aligned}
\Phi_\text{S} - \Phi_\text{U} &= k\left[-(z_{2\text{U}} - z_{1\text{S}}) + (z_{4\text{S}} - z_{3\text{U}})\right] \\
&= k\left[-\left(v_1 T - \tfrac{1}{2} g\, T^2\right) + \left((v_1 + 2v_\text{r})T - \tfrac{1}{2} g\,(3T^2 + TT')\right)\right] \\
&= k\Big[+2v_\text{r} - g\,(T + T')\Big]T \qquad (2.26)
\end{aligned}$$

Compare this result with the same expression from interferometer $\boxed{1}$ in equation (2.11) and note that the term proportional to v_r changes sign while the term proportional to g does not.

To evaluate the $-\omega t$ term, we calculate the resonance condition for each $\pi/2$-pulse using the velocity in equation (2.22) and again assuming an offset δ.

$$\omega_1 - \omega_\mathrm{ab} = \delta + \bar{k}_1 \bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} = \delta + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \tag{2.27}$$

$$\omega_2 - \omega_\mathrm{ab} = \delta + \bar{k}_2 \bar{v}_2 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} = \delta + \bar{k}\left[\bar{v}_1 - \bar{g}\,T\right] + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \tag{2.28}$$

$$\begin{aligned} \omega_3 - \omega_\mathrm{ab} &= \delta + \bar{k}_3 \bar{v}_3 - \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} = \delta - \bar{k}\left[\bar{v}_1 + \bar{v}_\mathrm{r} - \bar{g}\,(T+T')\right] - \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \\ &= \delta + \bar{k}\left[-\bar{v}_1 - \tfrac{3}{2}\bar{v}_\mathrm{r} + \bar{g}\,(T+T')\right] \end{aligned} \tag{2.29}$$

$$\begin{aligned} \omega_4 - \omega_\mathrm{ab} &= \delta + \bar{k}_4 \bar{v}_4 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} = \delta - \bar{k}\left[\bar{v}_1 + 2\bar{v}_\mathrm{r} - \bar{g}\,(2T+T')\right] + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \\ &= \delta + \bar{k}\left[-\bar{v}_1 - \tfrac{3}{2}\bar{v}_\mathrm{r} + \bar{g}\,(2T+T')\right] \end{aligned} \tag{2.30}$$

Using the rules in Table 2.1, one can show that interferometers $\boxed{1}$ and $\boxed{2}$ have the exact same sensitivity to the $-\omega t$ term: $+\phi_1(\omega) - \phi_2(\omega) + \phi_3(\omega) - \phi_4(\omega)$. For the shifted path, the first $\pi/2$-pulse adds and the fourth pulse subtracts. For the unshifted path, the second pulse adds and the third pulse subtracts. Thus, the evaluation of the total phase contribution from this term is the same as in equation (2.16).

$$\begin{aligned} \Phi_\mathrm{S} - \Phi_\mathrm{U} &= -[\Delta\phi_{12}(\omega)] - [\Delta\phi_{34}(\omega)] \\ &= -[-\omega_1\,(t_{12}-t_1) - \omega_2\,(t_2 - t_{12})] - [-\omega_3\,(t_{34}-t_3) - \omega_4\,(t_4 - t_{34})] \\ &= \left[\omega_\mathrm{ab} + \delta + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r}\right](t_{12} - t_1) \\ &\quad + \left[\omega_\mathrm{ab} + \delta + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} - \bar{k}\bar{g}\,T\right](t_2 - t_{12}) \\ &\quad + \left[\omega_\mathrm{ab} + \delta - \bar{k}\bar{v}_1 - \tfrac{3}{2}\bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(T+T')\right](t_{34} - t_3) \\ &\quad + \left[\omega_\mathrm{ab} + \delta - \bar{k}\bar{v}_1 - \tfrac{3}{2}\bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(2T+T')\right](t_4 - t_{34}) \\ &= \left[\omega_\mathrm{ab} + \delta + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r}\right]T - \bar{k}\bar{g}\,T\,(t_2 - t_{12}) \\ &\qquad + \left[\omega_\mathrm{ab} + \delta - \bar{k}\bar{v}_1 - \tfrac{3}{2}\bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(T+T')\right]T + \bar{k}\bar{g}\,T\,(t_4 - t_{34}) \\ &= \left[2(\omega_\mathrm{ab} + \delta) - \bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(T+T')\right]T + \bar{k}\bar{g}\,T\left[(t_4 - t_2) - (t_{34} - t_{12})\right] \end{aligned}$$

$$= \Big[2(\omega_{\text{ab}} + \delta) - \bar{k}\bar{v}_{\text{r}} + \bar{k}\bar{g}\,(T + T')\Big]\, T \tag{2.31}$$

Here we have inserted frequencies from equations (2.27) through (2.30) and again used the same constraints on the timing of the $\pi/2$-pulses discussed in the previous section. Comparing this result with the same expression from interferometer $\boxed{1}$ in equation (2.18) again reveals that the term proportional to v_{r} changes sign while the term proportional to g does not. Any additional optical phase (the ϕ term) also has the same expression: $\phi_1 - \phi_2 + \phi_3 - \phi_4$ as we had for $\boxed{1}$ in equation (2.19).

Combining equations (2.26) and (2.31), we can now construct the interferometer phase $\Phi(\text{light})$ for interferometer geometry $\boxed{2}$.

$$\begin{aligned}
&\Phi_{\boxed{2}}(\text{light}) \\
&= k\,[+2v_{\text{r}} - g\,(T + T')]\,T + \Big[2(\omega_{\text{ab}} + \delta) - \bar{k}\bar{v}_{\text{r}} + \bar{k}\bar{g}\,(T + T')\Big]\,T \\
&= \Big[+kv_{\text{r}} + (kv_{\text{r}} - \bar{k}\bar{v}_{\text{r}}) - (kg - \bar{k}\bar{g})(T + T')\Big]\,T + 2(\omega_{\text{ab}} + \delta)T
\end{aligned} \tag{2.32}$$

Combining this result with the total phase shift from the free evolution of the atom between the pulses in equation (2.25), we have the complete phase difference for interferometer geometry $\boxed{2}$.

$$\begin{aligned}
\Phi_{\boxed{2}} &= \Phi_{\boxed{2}}(\text{atom}) + \Phi_{\boxed{2}}(\text{light}) \\
&= -\frac{m v_{\text{r}} T}{\hbar} v_{\text{r}} - 2\omega_{\text{ab}} T \\
&\qquad + \Big[+kv_{\text{r}} + (kv_{\text{r}} - \bar{k}\bar{v}_{\text{r}}) - (kg - \bar{k}\bar{g})(T + T')\Big]\,T + 2(\omega_{\text{ab}} + \delta)T \\
&= \Big[+(kv_{\text{r}} - \bar{k}\bar{v}_{\text{r}}) - (kg - \bar{k}\bar{g})(T + T')\Big]\,T + 2\delta T
\end{aligned} \tag{2.33}$$

We are now ready to describe the result of a single measurement of the recoil shift. Because of its sensitivity to the local gravitational acceleration and any arbitrary offset δ of the laser frequencies, a measurement of the phase shift for interferometer geometry $\boxed{1}$ alone is not sufficient to measure the recoil shift. However, by measuring the phase shift from geometries $\boxed{1}$ and $\boxed{2}$ and subtracting the result in equation (2.21) from that in equation (2.33), we have a result that is sensitive only to the difference between the accepted and actual values of the recoil shift.

$$\begin{aligned}\Phi_{\boxed{1}} - \Phi_{\boxed{2}} &= \Big\{[-(kv_{\rm r} - \bar{k}\bar{v}_{\rm r}) - (kg - \bar{k}\bar{g})(T + T')]T + 2\delta T\Big\} \\ &\quad -\Big\{[+(kv_{\rm r} - \bar{k}\bar{v}_{\rm r}) - (kg - \bar{k}\bar{g})(T + T')]T + 2\delta T\Big\} \\ &= -2(kv_{\rm r} - \bar{k}\bar{v}_{\rm r})T \end{aligned} \tag{2.34}$$

This process is identical to the hypothetical measurement described in the introduction, except that interferometers are used to greatly increase the precision with which the center of the resonances can be determined.

2.1.3 π-pulses

To further improve the measurement sensitivity, we insert N π-pulses between the second and third $\pi/2$-pulses. These π-pulses are evenly spaced, separated by a time $T_{\pi\pi}$. Each π-pulse simultaneously addresses both interferometer paths. Because the atoms in each interferometer path are all in either state $|\text{a}\rangle$ or state $|\text{b}\rangle$, the π-pulse transfers the atoms in both paths to the other state, $|\text{a}\rangle$ to $|\text{b}\rangle$, and *vice versa*. When the atoms change state they undergo a stimulated absorption (and emission) and recoil accordingly. After each π-pulse, we change the beam direction so that the atoms continue recoiling in one direction. For the normal interferometers ($\boxed{1}$ and $\boxed{2}$), the π-pulses start with $\mathbf{k}_{\rm eff} = -k_{\rm eff}\hat{\mathbf{z}}$. Interferometer $\boxed{1}$ shown in Figure 2.3 for $N = 2$ leaves the atoms in the state $|\text{a}\rangle$ after the second $\pi/2$-pulse. These atoms recoil downward, and end in state $|\text{b}\rangle$. For the second π-pulse, the beam direction reverses. However, because the atoms are now in state $|\text{b}\rangle$, the recoil is again downward. For interferometer $\boxed{2}$ shown in Figure 2.3, the atoms leave the second $\pi/2$-pulse in state $|\text{b}\rangle$. Consequently, each π-pulse causes the atoms to recoil upward. Since the π-pulses push interferometer $\boxed{1}$ down, we will often refer to it as a "down" interferometer. Its conjugate interferometer which is pushed upward by the π-pulses will be called an "up" interferometer.

To determine how the π-pulses change the final interferometer phase, we go through the same process we completed in Sections 2.1.1 2.1.2 for the $\pi/2$-pulses

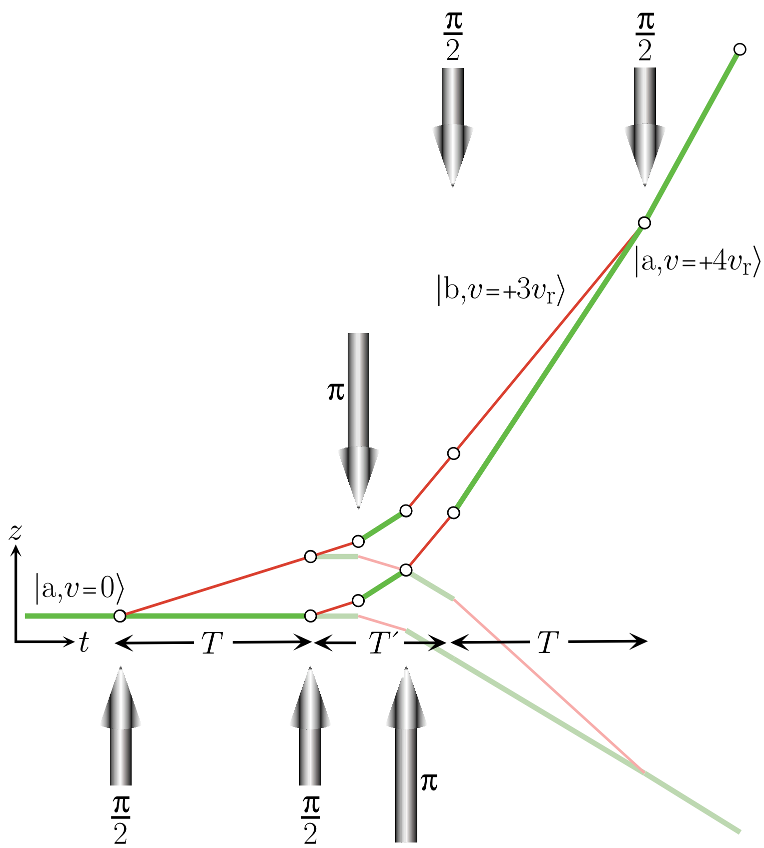


Figure 2.3: **Recoil measurement with two additional recoils** ($N = 2$)**.** By inserting N π-pulses between the second and third $\pi/2$-pulses, the separation between the two conjugate interferometers can be increased by N additional recoils. Because the π-pulses do not affect the fringe spacing, they do not change the precision with which we can determine the centers of the two resonances. Thus, by adding addition recoils, we linearly increase the final measurement resolution. Since we know the integer number N, we divide the final frequency difference by $(N+1)$ to derive the size of a single recoil.

but this time for π-pulses. In addition, due to the π-pulses, the input velocity and position for these pulses is now different, so we must modify the results of these sections for the last two $\pi/2$-pulses.

N even

If the number of π-pulses N is odd, the atoms in both interferometer paths will enter the third $\pi/2$-pulse in the other state. Although, a perfectly viable interferometer results, it significantly changes the final phase expression, so for now we will assume that N is even, as shown in Figure 2.3. We first calculate the velocity and position of atoms for the shifted (S) and unshifted (U) paths of interferometer $\boxed{1}$. For both paths, the velocity $v_{\pi i}$ before the ith π-pulse will be

$$\begin{aligned}
v_{\pi 1} &= v_{2\mathrm{U}} - g(t_{\pi 1} - t_2) = v_{2\mathrm{U}} - gT_{\pi\pi} \\
v_{\pi 2} &= v_{\pi 1} - g(t_{\pi 2} - t_{2\pi 1}) - v_{\mathrm{r}} = v_{2\mathrm{U}} - 2gT_{\pi\pi} - v_{\mathrm{r}} \\
v_{\pi 3} &= v_{\pi 2} - g(t_{\pi 3} - t_{2\pi 2}) - v_{\mathrm{r}} = v_{2\mathrm{U}} - 3gT_{\pi\pi} - 2v_{\mathrm{r}} \\
&\vdots \\
v_{\pi N} &= v_{2\mathrm{U}} - NgT_{\pi\pi} - (N-1)v_{\mathrm{r}}
\end{aligned} \tag{2.35}$$

For the unshifted path, the position $z_{\pi i\mathrm{U}}$ at the ith π-pulse will be

$$\begin{aligned}
z_{\pi 1\mathrm{U}} &= z_{2\mathrm{U}} + v_1(t_{\pi 1} - t_2) - \tfrac{1}{2}g(t_{\pi 1}^2 - t_2^2) \\
&= z_{2\mathrm{U}} + v_1 T_{\pi\pi} - \tfrac{1}{2}gT_{\pi\pi}(2T + T_{\pi\pi}) \\
z_{\pi 2\mathrm{U}} &= z_{\pi 1\mathrm{U}} + [v_1 - v_{\mathrm{r}}](t_{\pi 2} - t_{\pi 1}) - \tfrac{1}{2}g(t_{\pi 2}^2 - t_{\pi 1}^2) \\
&= z_{2\mathrm{U}} + [2v_1 - v_{\mathrm{r}}]T_{\pi\pi} - \tfrac{1}{2}gT_{\pi\pi}(4T + 4T_{\pi\pi}) \\
z_{\pi 3\mathrm{U}} &= z_{\pi 2\mathrm{U}} + [v_1 - 2v_{\mathrm{r}}](t_{\pi 3} - t_{\pi 2}) - \tfrac{1}{2}g(t_{\pi 3}^2 - t_{\pi 2}^2) \\
&= z_{2\mathrm{U}} + [3v_1 - 3v_{\mathrm{r}}]T_{\pi\pi} - \tfrac{1}{2}gT_{\pi\pi}(6T + 9T_{\pi\pi}) \\
&\vdots \\
z_{\pi N\mathrm{U}} &= z_{2\mathrm{U}} + \left[Nv_1 - \tfrac{1}{2}N(N-1)v_{\mathrm{r}}\right]T_{\pi\pi} - \tfrac{1}{2}gT_{\pi\pi}(2NT + N^2T_{\pi\pi})
\end{aligned} \tag{2.36}$$

At each π-pulse, the shifted path is a fixed displacement $\Delta z = +v_{\mathrm{r}}T$ from the unshifted path

$$z_{\pi i\mathrm{S}} = z_{\pi i\mathrm{U}} + v_{\mathrm{r}}T \tag{2.37}$$

The velocity given in equation (2.5) for third and fourth $\pi/2$-pulse must be modified to include the term $-Nv_{\mathrm{r}}$ from the π-pulses

$$\begin{aligned}
v_{3\mathrm{U}} &= v_1 - g\,(T + T') - Nv_{\mathrm{r}} & \quad v_{3\mathrm{S}} &= v_1 - g\,(T + T') - Nv_{\mathrm{r}} \\
v_{4\mathrm{U}} &= v_1 - g\,(2T + T') - Nv_{\mathrm{r}} & \quad v_{4\mathrm{S}} &= v_1 - g\,(2T + T') - (N+1)v_{\mathrm{r}}
\end{aligned} \tag{2.38}$$

Similarly, the positions in equation (2.6) for the unshifted and shifted paths become

$$\left.\begin{aligned} z_{3\mathrm{U}} &= z_{\pi N\mathrm{U}} + [v_1 - Nv_\mathrm{r}](T' - NT_{\pi\pi}) - \tfrac{1}{2}g\Big[(T+T')^2 - (T+NT_{\pi\pi})^2\Big] \\ &= z_{2\mathrm{U}} + [v_1 - Nv_\mathrm{r}]T' + \tfrac{1}{2}N(N+1)v_\mathrm{r}T_{\pi\pi} - \tfrac{1}{2}g\Big[2TT' + T'^2\Big] \\ z_{4\mathrm{U}} &= z_{3\mathrm{U}} + [v_1 - Nv_\mathrm{r}]T - \tfrac{1}{2}g\Big[3T^2 + 2TT'\Big] \end{aligned}\right| \\ \left| \begin{aligned} z_{3\mathrm{S}} &= z_{2\mathrm{S}} + [v_1 - Nv_\mathrm{r}]T' + \tfrac{1}{2}N(N+1)v_\mathrm{r}T_{\pi\pi} - \tfrac{1}{2}g\Big[2TT' + T'^2\Big] \\ z_{4\mathrm{S}} &= z_{3\mathrm{S}} + [v_1 - (N+1)v_\mathrm{r}]T - \tfrac{1}{2}g\Big[3T^2 + 2TT'\Big] \end{aligned}\right. \tag{2.39}$$

As before, with the position at each pulse we are now prepared to evaluate the action over each path and compute the phase difference $\Delta S_\mathrm{cl}/\hbar = (S_\mathrm{cl}(\text{shifted}) - S_\mathrm{cl}(\text{unshifted})/\hbar$. Because the displacement between the two interferometer paths between the second and third $\pi/2$-pulses is not affected by the π-pulses, equation (2.8) for the phase due the action does not change. Equation (2.9), however, for the phase due to the action between the last two $\pi/2$-pulses does change.

$$\begin{aligned} &\Delta S_\mathrm{cl}(34)/\hbar \\ &= \frac{m}{2\hbar}\Bigg\{\frac{\Big(v_1 - (\underbrace{N}+1)v_\mathrm{r}]T - \tfrac{1}{2}g\,T(3T+2T')\Big)^2}{T} \\ &\qquad - \frac{\Big([v_1 - \underbrace{Nv_\mathrm{r}}]T - \tfrac{1}{2}g\,T(3T+2T')\Big)^2}{T} - g\,T\,[v_\mathrm{r}T]\Bigg\} - \Big[(\omega_\mathrm{b} - \omega_\mathrm{a})T\Big] \\ &= \frac{m}{2\hbar}\Big\{-2v_1v_\mathrm{r}T + \underbrace{2Nv_\mathrm{r}^2T} + v_\mathrm{r}^2T + v_\mathrm{r}g\,T(3T+2T') - v_\mathrm{r}g\,T^2\Big\} - \omega_\mathrm{ab}T \\ &= \frac{m}{2\hbar}v_\mathrm{r}T\Big\{-2v_1 + (\underbrace{2N}+1)v_\mathrm{r} + 2g(T+T')\Big\} - \omega_\mathrm{ab}T \end{aligned} \tag{2.40}$$

where the underbraces ($\underbrace{\quad}$) indicate new terms due to the π-pulses. Summing equations (2.7), (2.8), and (2.40), we have the total phase difference accumulated from the evolution of the atomic wavefunction along both paths

$$\begin{aligned} \Phi_{\boxed{1}}(\text{atom}) &= [\Delta S_\mathrm{cl}(12) + \Delta S_\mathrm{cl}(23) + \Delta S_\mathrm{cl}(34)]/\hbar \\ &= \frac{m}{2\hbar}v_\mathrm{r}T\{\underbrace{2Nv_\mathrm{r}} + 2v_\mathrm{r}\} - 2\omega_\mathrm{ab}T \\ &= \frac{mv_\mathrm{r}^2T}{\hbar}(\underbrace{N}+1) - 2\omega_\mathrm{ab}T \end{aligned} \tag{2.41}$$

We next derive the terms for the contribution from the laser at each pulse. As in Section 2.1.1, we consider the kz, $-\omega t$, and ϕ components of the optical phase separately.

kz term

At each π-pulse, the atomic state changes in both interferometer paths. Thus, the contribution from the π-pulses for the shifted and unshifted paths is

$$\Phi_{\pi\mathrm{U}} = k_{\pi 1} z_{\pi 1\mathrm{U}} - k_{\pi 2} z_{\pi 2\mathrm{U}} + k_{\pi 3} z_{\pi 3\mathrm{U}} - \cdots - k_{\pi N} z_{\pi N\mathrm{U}} \quad (2.42)$$

$$\Phi_{\pi\mathrm{S}} = k_{\pi 1} z_{\pi 1\mathrm{S}} - k_{\pi 2} z_{\pi 2\mathrm{S}} + k_{\pi 3} z_{\pi 3\mathrm{S}} - \cdots - k_{\pi N} z_{\pi N\mathrm{S}} \quad (2.43)$$

where the sign of each term is determined by the state of the atoms at the beginning of the π-pulse: positive for $|\mathrm{a}\rangle \to |\mathrm{b}\rangle$ and negative for $|\mathrm{b}\rangle \to |\mathrm{a}\rangle$. The difference between the shifted and unshifted paths is

$$\Phi_{\pi\mathrm{S}} - \Phi_{\pi\mathrm{U}} = k_{\pi 1}\Delta z_{\pi 1} - k_{\pi 2}\Delta z_{\pi 2} + k_{\pi 3}\Delta z_{\pi 3} - \cdots - k_{\pi N}\Delta z_{\pi N} \quad (2.44)$$

where $\Delta z_{\pi i} = \Delta z = v_{\mathrm{r}} T$ is the constant separation between the two interferometer paths for the whole time between the second and third $\pi/2$-pulses. Since the beam direction alternates for each pulse starting with $k_{\pi 1} = -k$ and the recoil size for each π-pulse is the same, equation (2.44) becomes

$$\begin{aligned} \Phi_{\pi\mathrm{S}} - \Phi_{\pi\mathrm{U}} &= (-k - k - k - \cdots - k)\Delta z \\ &= -Nkv_{\mathrm{r}}T \end{aligned} \quad (2.45)$$

For the first and second $\pi/2$-pulses the kz-expression remains the same, because the atom's position for these pulses is unchanged. For the last two $\pi/2$-pulses, however, because of the velocity added by the π-pulses, the distance $z_4 - z_3$ of the atom's position between the third and fourth $\pi/2$-pulse has an additional term $-Nv_{\mathrm{r}}T$. With the addition of equation (2.45), equation (2.11) becomes

$$\begin{aligned}
\Phi_\mathrm{S} - \Phi_\mathrm{U} &= \underbrace{\Phi_{\pi\mathrm{S}} - \Phi_{\pi\mathrm{U}}} + k\left[-(z_{2\mathrm{S}} - z_{1\mathrm{S}}) + (z_{4\mathrm{S}} - z_{3\mathrm{S}})\right] \\
&= \underbrace{-Nkv_\mathrm{r}T} + k\Big[-\Big((v_1 + v_\mathrm{r})T - \tfrac{1}{2}g\,T^2\Big) \\
&\qquad + \Big((v_1 - (\underbrace{N} + 1)v_\mathrm{r})T - \tfrac{1}{2}g\,(3T^2 + 2TT')\Big)\Big] \\
&= k\Big[-2(\underbrace{N}+1)v_\mathrm{r} - g\,(T+T')\Big]T \qquad (2.46)
\end{aligned}$$

$-\omega t$ term

To calculate the phase for the time varying component of the optical phase, we must know the frequency difference between the lasers for each pulse. Since this frequency difference is always the same for both interferometer paths, each π-pulse adds the same phase to each path and thus no net phase is contributed to the $-\omega t$ term. For the first two $\pi/2$-pulses, the frequencies are the same. For the last two $\pi/2$-pulses, equations (2.14) and (2.15) become

$$\begin{aligned}
\omega_3 - \omega_\mathrm{ab} &= \delta + \bar{k}_3\bar{v}_3 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \\
&= \delta - \bar{k}\left[\bar{v}_1 - \bar{g}\,(T+T')\right] + (\underbrace{N} + \tfrac{1}{2})\bar{k}\bar{v}_\mathrm{r} \qquad (2.47) \\
\omega_4 - \omega_\mathrm{ab} &= \delta + \bar{k}_4\bar{v}_4 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r} \\
&= \delta - \bar{k}\left[\bar{v}_1 - \bar{g}\,(2T+T')\right] + (\underbrace{N} + \tfrac{1}{2})\bar{k}\bar{v}_\mathrm{r} \qquad (2.48)
\end{aligned}$$

so that equation (2.18) for the total contribution from the $-\omega t$ term of the optical phase for interferometers with N π-pulses (with N even) becomes

$$\begin{aligned}
\Phi_\mathrm{S} - \Phi_\mathrm{U} &= \Big[\delta + \omega_\mathrm{ab} + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_\mathrm{r}\Big]T - \bar{k}\bar{g}\,T\,(t_2 - t_{12}) \\
&\qquad + \Big[\delta + \omega_\mathrm{ab} - \bar{k}\bar{v}_1 + (\underbrace{N} + \tfrac{1}{2})\bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(T+T')\Big]T + \bar{k}\bar{g}\,T\,(t_4 - t_{34}) \\
&= \Big[2(\omega_\mathrm{ab} + \delta) + (\underbrace{N} + 1)\bar{k}\bar{v}_\mathrm{r} + \bar{k}\bar{g}\,(T+T')\Big]T \qquad (2.49)
\end{aligned}$$

ϕ term

Just as the π-pulses did not contribute a net phase to the $-\omega t$ term, they also contribute no net phase to the static phase term. Equation (2.19) still applies, independent of N.

Combining equations (2.41), (2.46), and (2.49), we can now evaluate the complete interferometer phase difference $\Phi_{\boxed{1}}$, including the evolution of the atomic wavefunction and the contribution from the optical phase being imposed on the atoms at each pulse. Analogous to equation (2.21), we have

$$\begin{aligned} \Phi_{\boxed{1}} &= \Phi_{\boxed{1}}(\text{atom}) + \Phi_{\boxed{1}}(\text{light}) \\ &= \frac{mv_\mathrm{r}T}{\hbar}(N+1)v_\mathrm{r} - 2\omega_\mathrm{ab}T \\ &\quad +(N+1)\left[-kv_\mathrm{r} - (kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r})\right]T - \left[(kg - \bar{k}\bar{g})(T+T')\right]T + 2(\omega_\mathrm{ab}+\delta)T \\ &= \left[-(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2\delta T \end{aligned} \tag{2.50}$$

and analogous to equation (2.33) for the conjugate interferometer pushed up by the π-pulses, we have

$$\begin{aligned} \Phi_{\boxed{2}} &= \Phi_{\boxed{2}}(\text{atom}) + \Phi_{\boxed{2}}(\text{light}) \\ &= -\frac{mv_\mathrm{r}T}{\hbar}(N+1)v_\mathrm{r} - 2\omega_\mathrm{ab}T \\ &\quad +(N+1)\left[+kv_\mathrm{r} + (kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r})\right]T - \left[(kg - \bar{k}\bar{g})(T+T')\right]T + 2(\omega_\mathrm{ab}+\delta)T \\ &= \left[+(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) - (kg - \bar{k}\bar{g})(T+T')\right]T + 2\delta T \end{aligned} \tag{2.51}$$

The difference between the two measurements (2.50) and (2.51) is the measurement of the recoil frequency. Analogous to equation (2.34), we now have

$$\begin{aligned} \Phi_{\boxed{1}} - \Phi_{\boxed{2}} &= \left\{[-(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) - (kg - \bar{k}\bar{g})(T+T')]T + 2\delta T\right\} \\ &\quad -\left\{[+(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) - (kg - \bar{k}\bar{g})(T+T')]T + 2\delta T\right\} \\ &= -2(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r})T \end{aligned} \tag{2.52}$$

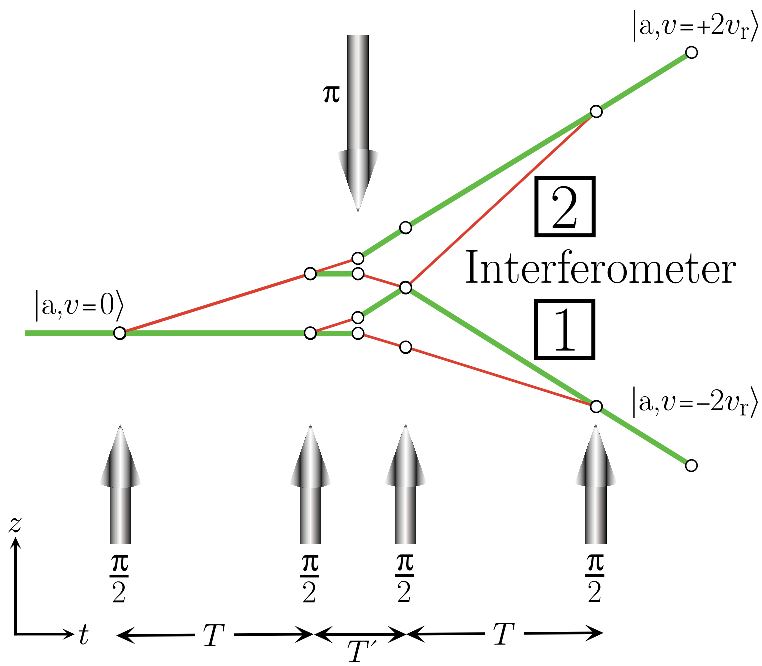


Figure 2.4: **Recoil measurement with one additional recoil** ($N = 1$)**.** When N is odd, the beam direction for the final two $\pi/2$-pulses must reverse (compare with Figure 2.3). Otherwise, because the atomic state entering the third $\pi/2$-pulse is reversed, the final two $\pi/2$-pulses would cancel the effect of the first two $\pi/2$-pulses and thereby reduce the frequency separation by one recoil frequency. To maintain the frequency separation between the interferometers proportional to $N + 1$, we therefore reverse the direction of the lasers for the final two $\pi/2$-pulses and correct their frequencies accordingly.

N odd

Interferometers constructed with an odd number of π-pulses enter the third $\pi/2$-pulse in the opposite state they exit the second $\pi/2$-pulse. Figure 2.4 shows an example with $N = 1$. As a result, the roles of the states $|a\rangle$ and $|b\rangle$ are reversed for the last $\pi/2$-pulse pair. In all of the different expressions which contribute to the final interferometer phase difference, when N is even, the first and second $\pi/2$-pulse pair produce terms proportional to the recoil shift with the same sign, thus doubling that term in the final expression. For example, in equation (2.11) the first and second $\pi/2$-pulse pair both generate the term $-kv_{\mathrm{r}}T$, which adds up to $-2kv_{\mathrm{r}}T$ in total. On the other hand, when N is odd, with all of the beam directions unchanged the terms proportional to the recoil frequency from the second $\pi/2$-pulse pair have the opposite sign compared to the terms from the first $\pi/2$-pulse pair. In all of the different components of the final phase, the terms from the two $\pi/2$-pulse pairs cancel, reducing the number of recoils from $N+1$ to N. To avoid reducing the number of recoils, when N is odd we reverse the beam direction for the final two $\pi/2$-pulses. Thus, the kz term for each of the four $\pi/2$-pulses contributes as

Geometry $\boxed{1}$	$\pi/2$-pulse #1	#2	#3	#4
Direction of $\lvert\mathbf{k}\rvert$	Up	Up	Up	Up
Unshifted path	0	0	0	$-k_4 z_{4\mathrm{U}}$ $= -kz_{4\mathrm{U}}$
Shifted path	$+k_1 z_{1\mathrm{S}}$ $= +kz_{1\mathrm{S}}$	$-k_2 z_{2\mathrm{S}}$ $= -kz_{2\mathrm{S}}$	$-k_3 z_{3\mathrm{S}}$ $= -kz_{3\mathrm{S}}$	0

Although we now have a contribution from the unshifted path, each term still contributes with the sign convention given in Table 2.1. In the shifted path the kz term adds when the atom changes from state $|\mathrm{a}\rangle$ to $|\mathrm{b}\rangle$ at the first $\pi/2$-pulse and subtracts when the atom goes from $|\mathrm{b}\rangle$ to $|\mathrm{a}\rangle$ at the second and third $\pi/2$-pulses. At the last $\pi/2$-pulse in the unshifted path, the kz term again subtracts when the atom changes from $|\mathrm{b}\rangle$ to $|\mathrm{a}\rangle$. Since the positions in equation (2.39) still apply, the total contribution from the kz term is thus identical to equation (2.46).

The total phase difference from the atom's evolution in the dark and from the remaining terms in the optical phase do change, however. Because the roles of the atomic states are reversed for the final two $\pi/2$-pulses, the phase evolution due to the energy difference $\hbar\omega_{\mathrm{ab}}$ between the states exactly cancels the analogous term from the first two $\pi/2$-pulses. The difference of the total action phase between the two interferometer paths is thus

$$\Phi_{\boxed{1}}(\mathrm{atom}) = \frac{mv_{\mathrm{r}}^2 T}{\hbar}(N+1) \tag{2.53}$$

the same as equation (2.41) but without the $-2\omega_{\mathrm{ab}}T$ term.

The contribution from the static phase of the lasers is no longer given by equation (2.19). Instead

$$\Phi_{\rm S} - \Phi_{\rm U} = \phi_1 - \phi_2 \underbrace{-\phi_3 + \phi_4} \tag{2.54}$$

where the underbrace indicates the terms whose sign is reversed. Similarly, the phase for the $-\omega t$ term is now

$$\Phi_{\rm S} - \Phi_{\rm U} \;=\; +\phi_1(\omega) - \phi_2(\omega) - \phi_3(\omega) + \phi_4(\omega) \;=\; -\Delta\phi_{12}(\omega) \underbrace{+} \Delta\phi_{34}(\omega) \tag{2.55}$$

Because the input state is different and the beam direction is no longer reversed for the final two $\pi/2$-pulses, the frequencies for these pulses given in equations (2.47) and (2.48) become

$$\begin{aligned}
\omega_3 - \omega_{\rm ab} &= \delta + \bar{k}_3\bar{v}_3 - \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} \\
&= \delta + \bar{k}\left[\bar{v}_1 - \bar{g}\,(T+T')\right] - (N+\tfrac{1}{2})\bar{k}\bar{v}_{\rm r} && (2.56) \\
\omega_4 - \omega_{\rm ab} &= \delta + \bar{k}_4\bar{v}_4 - \tfrac{1}{2}\bar{k}\bar{v}_{\rm r} \\
&= \delta + \bar{k}\left[\bar{v}_1 - \bar{g}\,(2T+T')\right] - (N+\tfrac{1}{2})\bar{k}\bar{v}_{\rm r} && (2.57)
\end{aligned}$$

so that equation (2.55) becomes

$$\begin{aligned}
\Phi_{\rm S} - \Phi_{\rm U} &= \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r}\right](t_{12} - t_1) \\
&\quad + \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 - \bar{k}\bar{g}\,T + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r}\right](t_2 - t_{12}) \\
&\quad - \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 - \bar{k}\bar{g}\,(T+T') - (N+\tfrac{1}{2})\bar{k}\bar{v}_{\rm r}\right](t_{34} - t_3) \\
&\quad - \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 - \bar{k}\bar{g}\,(2T+T') - (N+\tfrac{1}{2})\bar{k}\bar{v}_{\rm r}\right](t_4 - t_{34}) \\
&= \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 + \tfrac{1}{2}\bar{k}\bar{v}_{\rm r}\right]T - \bar{k}\bar{g}\,T\,(t_2 - t_{12}) \\
&\qquad - \left[\delta + \omega_{\rm ab} + \bar{k}\bar{v}_1 - (N+\tfrac{1}{2})\bar{k}\bar{v}_{\rm r} - \bar{k}\bar{g}\,(T+T')\right]T + \bar{k}\bar{g}\,T\,(t_4 - t_{34}) \\
&= \left[(N+1)\bar{k}\bar{v}_{\rm r} + \bar{k}\bar{g}\,(T+T')\right]T + \bar{k}\bar{g}\,T\left[(t_4 - t_2) - (t_{34} - t_{12})\right] \\
&= \left[(N+1)\bar{k}\bar{v}_{\rm r} + \bar{k}\bar{g}\,(T+T')\right]T && (2.58)
\end{aligned}$$

which is the same as equation (2.49) without the $2(\omega_{\mathrm{ab}}+\delta)T$ term.

Combining equations (2.46), (2.53), and (2.58), we have the complete expression for interferometer $\boxed{1}$ with N odd

$$\begin{aligned}
&\Phi_{\boxed{1}}(N \text{ odd}) \\
&\quad = \frac{m v_{\mathrm{r}} T}{\hbar}(N+1) v_{\mathrm{r}} \\
&\qquad +(N+1)\left[-k v_{\mathrm{r}}-(k v_{\mathrm{r}}-\bar{k} \bar{v}_{\mathrm{r}})\right] T-\left[(k g-\bar{k} \bar{g})(T+T')\right] T \\
&\quad = \left[-(N+1)(k v_{\mathrm{r}}-\bar{k} \bar{v}_{\mathrm{r}})-(k g-\bar{k} \bar{g})(T+T')\right] T
\end{aligned} \tag{2.59}$$

which is identical to equation (2.50) without the $+2\delta T$ term. Similarly, equation (2.51) for the conjugate interferometer $\boxed{2}$ becomes

$$\begin{aligned}
&\Phi_{\boxed{2}}(N \text{ odd}) \\
&\quad = -\frac{m v_{\mathrm{r}} T}{\hbar}(N+1) v_{\mathrm{r}}-2 \omega_{\mathrm{ab}} T \\
&\qquad +(N+1)\left[+k v_{\mathrm{r}}+(k v_{\mathrm{r}}-\bar{k} \bar{v}_{\mathrm{r}})\right] T-\left[(k g-\bar{k} \bar{g})(T+T')\right] T \\
&\quad = \left[+(N+1)(k v_{\mathrm{r}}-\bar{k} \bar{v}_{\mathrm{r}})-(k g-\bar{k} \bar{g})(T+T')\right] T
\end{aligned} \tag{2.60}$$

so that the up/down difference for N odd is identical to equation (2.52). Comparing the results from interferometers with N odd and N even is an important test for systematic errors (see Section 6.2.3).

2.1.4 Inverted Interferometers

Interferometer $\boxed{2}$ differs from its conjugate interferometer $\boxed{1}$ by selecting the other internal state after the second $\pi/2$-pulse. If we now take these two "normal" interferometers and reverse the direction of all of the recoils, we can generate the final two interferometer geometries $\boxed{3}$ and $\boxed{4}$. As shown in Figure 2.5, interferometers $\boxed{3}$ and $\boxed{4}$ look the same as interferometers $\boxed{1}$ and $\boxed{2}$, respectively, but reflected about the horizontal axis. By measuring the phase from these two "inverted" interferometers, we are able to eliminate many systematic errors such as shifts due to magnetic bias

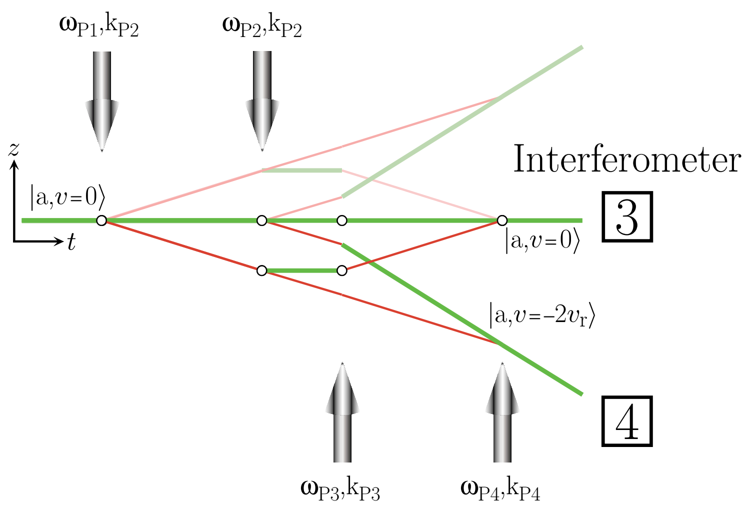


Figure 2.5: **Inverted interferometers**, generated by reversing the direction of all of the lasers for each of the pulses. These new interferometer geometries labeled $\boxed{3}$ and $\boxed{4}$ appear the same as the normal interferometers $\boxed{1}$ and $\boxed{2}$ (shown in gray) reflected about the horizontal axis. Except for the sign of the gravity term, the final phase expressions for $\boxed{1}$ and $\boxed{3}$ are the same, and similarly for $\boxed{2}$ and $\boxed{4}$. However, since interferometer $\boxed{3}$ is pushed *up* by the π-pulses while $\boxed{1}$ is pushed *down*, the roles of up and down are reversed for the inverted interferometers. This inversion helps cancel spatially dependent systematic effects such as those caused by the external magnetic field and the gravity gradient.

field inhomogeneities and gravity gradients that depend on where the interferometers occur in space. This cancelation occurs because interferometer $\boxed{3}$ for which the atom emerges from the second $\pi/2$-pulse in the $|\mathrm{a}\rangle$ state is pushed *up* by the π-pulses and has a phase expression

$$\Phi_{\boxed{3}} = \left[-(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) + (kg - \bar{k}\bar{g})(T+T')\right] T + 2\delta T \tag{2.61}$$

which is the same as the result in equation (2.50) for $\boxed{1}$, pushed *down* by the π-pulses, with the sign of the gravity term reversed. The roles of up and down are similarly reversed for interferometers $\boxed{2}$ and $\boxed{4}$.

$$\Phi_{\boxed{4}} = \left[+(N+1)(kv_\mathrm{r} - \bar{k}\bar{v}_\mathrm{r}) + (kg - \bar{k}\bar{g})(T+T')\right] T + 2\delta T \tag{2.62}$$

Combining the phase results from interferometers $\boxed{3}$ and $\boxed{4}$ in equations (2.61) and (2.62), we have

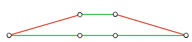
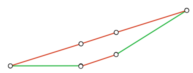
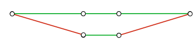
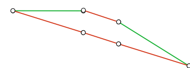
$$\Phi_{\boxed{3}} - \Phi_{\boxed{4}} = -2(N+1)(kv_{\mathrm{r}} - \bar{k}\bar{v}_{\mathrm{r}})T \tag{2.63}$$

which is identical to equation (2.52), the result for the normal interferometers $\boxed{1}$ and $\boxed{2}$. To combine the results from "normal" interferometers $\boxed{1}$ and $\boxed{2}$ with inverted interferometers $\boxed{3}$ and $\boxed{4}$, we calculate the unweighted arithmetic mean of equations (2.52) and (2.63)

$$\frac{1}{2}\left[\left(\Phi_{\boxed{1}} - \Phi_{\boxed{2}}\right) + \left(\Phi_{\boxed{3}} - \Phi_{\boxed{4}}\right)\right] = -2(N+1)(kv_{\mathrm{r}} - \bar{k}\bar{v}_{\mathrm{r}})T \tag{2.64}$$

The results for all four interferometer geometries are summarized in Table 2.2

Table 2.2: **The complete phase expression for the four fundamental interferometer geometries.** To calculate the phase difference between the two paths for each of the four fundamental interferometer geometries, we must evaluate first the phase evolution of the atomic wavefunction between the pulses when the light is off and second, the contribution of the optical phase $kz - \omega t + \phi$ imposed on the atom at each pulse.

	Interferometer Geometry			
	1 	3 	2 	4 
Recoil Direction	Normal	Inverted	Normal	Inverted
π-pulses push	Down	Up	Up	Down
Evolution of Atomic Wavefunction	$+\frac{mv_r^2 T}{\hbar}(N+1)\underbrace{-2\omega_{ab}T}$		$-\frac{mv_r^2 T}{\hbar}(N+1)\underbrace{-2\omega_{ab}T}$	
Light Pulses: kz term	$-2(N+1)kv_r T$ $-kg\,(T+T')T$		$+2(N+1)kv_r T$ $-kg\,(T+T')T$	
$-\omega t$ term	$\underbrace{2(\omega_{ab}+\delta)T}+(N+1)\bar{k}\bar{v}_r T$ $+\bar{k}\bar{g}\,(T+T')T$		$\underbrace{2(\omega_{ab}+\delta)T}-(N+1)\bar{k}\bar{v}_r T$ $+\bar{k}\bar{g}\,(T+T')T$	
ϕ term	$(\phi_1-\phi_2)+(-1)^N(\phi_3-\phi_4)$			
$\Phi(\boxed{1})$, $\Phi(\boxed{3})$	$-(N+1)(kv_r-\bar{k}\bar{v}_r)T$ $-(kg-\bar{k}\bar{g})\,(T+T')T$			
$\Phi(\boxed{2})$, $\Phi(\boxed{4})$			$+(N+1)(kv_r-\bar{k}\bar{v}_r)T$ $-(kg-\bar{k}\bar{g})\,(T+T')T$	
Up / Down Difference: $\Phi(\boxed{1})-\Phi(\boxed{2})$, $\Phi(\boxed{3})-\Phi(\boxed{4})$	$-2(N+1)(kv_r-\bar{k}\bar{v}_r)T$			

2.2 Two-photon transitions

2.2.1 Adiabatic passage

The ability to transfer atomic population from one state to another using adiabatic passage [16, 17, 18, 19] has been demonstrated in many different contexts [20, 21, 22, 23]. In this experiment we transfer cesium atoms between the $F=3$ and $F=4$ hyperfine ground states by adiabatically varying the intensity of two lasers tuned to the transitions from the ground states to the $F=3'$ excited state.

Consider the cesium atom energy structure shown in Figure 2.6 in the presence of two counter-propagating laser beams with frequencies ω_1 and ω_2 and wavevector magnitudes k_1 and k_2. We assume for now that the beams can be described by plane waves so that the electric field is

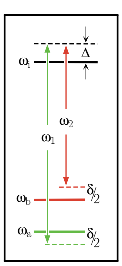
$$\mathbf{E}(z,t) = \mathbf{E}_1 \cos(k_1 z - \omega_1 t + \phi_1) + \mathbf{E}_2 \cos(-k_2 z - \omega_1 t + \phi_2) \tag{2.65}$$

Using the excited state $6\mathrm{P}_{1/2}(F=3)$ as intermediate state $|\mathrm{i}\rangle$, we can write the single-photon coupling strength in terms of Rabi frequencies

$$\Omega_1 = \frac{e}{\hbar}\left\langle \mathrm{i} \middle| \mathbf{r}\cdot\mathbf{E}_1 \middle| \mathrm{a} \right\rangle \tag{2.66}$$

$$\Omega_2 = \frac{e}{\hbar}\left\langle \mathrm{i} \middle| \mathbf{r}\cdot\mathbf{E}_2 \middle| \mathrm{b} \right\rangle \tag{2.67}$$

where $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$ represent the hyperfine ground states $6\mathrm{S}_{1/2}(F=3)$ and $6\mathrm{S}_{1/2}(F=4)$, respectively. The frequency detunings from resonance



$$\Delta_1 = \omega_1 - \omega_{\mathrm{ai}} = \Delta - \delta/2 - k_1 \cdot v - \frac{1}{2}\frac{\hbar}{m}k_1^2 \tag{2.68}$$

$$\Delta_2 = \omega_2 - \omega_{\mathrm{ai}} = \Delta + \delta/2 - k_1 \cdot v - \frac{1}{2}\frac{\hbar}{m}(k_1^2 + k_2^2) \tag{2.69}$$

can be defined in terms of a detuning Δ from the single-photon resonance condition and a relative detuning δ from the two-photon resonance.

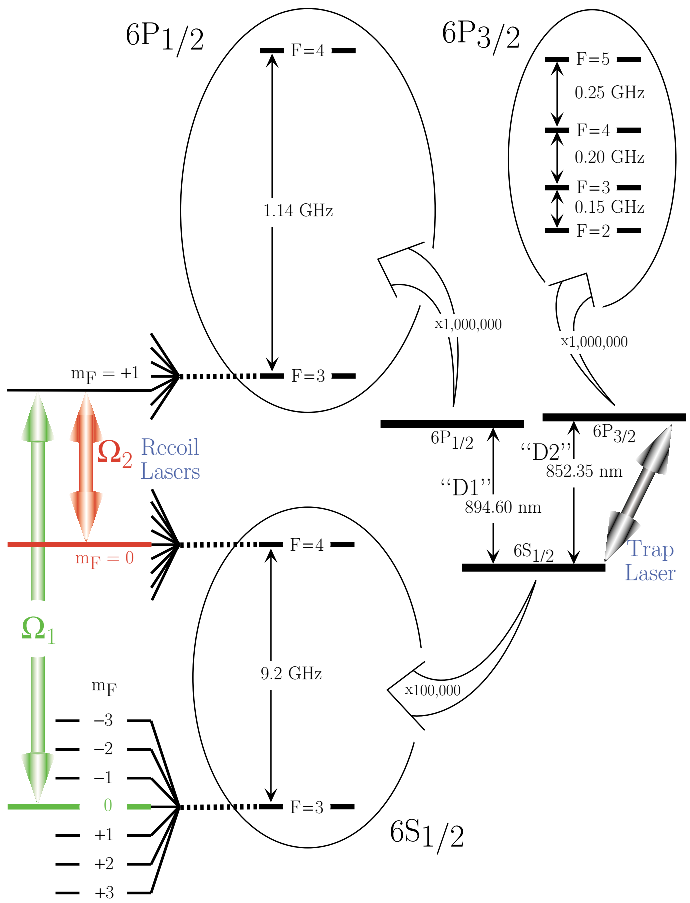


Figure 2.6: **The Cesium level structure** shown to scale. The $6S_{1/2}$ ground state couples via electric dipole transitions to the two fine structure split excited state levels $6P_{1/2}$ and $6P_{3/2}$. Because of the $I = \frac{7}{2}$ spin of the nucleus, each of these three levels is further split into $2J+1$ hyperfine levels, which can be further split into $2F+1$ magnetic sublevels in the presence of an external magnetic field. We use lasers tuned to the $6P_{3/2}$ excited state (often called the D2 transition) to cool and trap the atoms. To impart the recoils, build the interferometers, and make the measurement of ω_{rec}, we tune another set of lasers to the $6P_{1/2}$ excited state (D1 transition). These lasers drive two-photon transitions between the $F=3$ and $F=4$ hyperfine levels of the ground state. These two ground states are metastable and thus live forever on the time scales of our measurement. Because the frequency splitting of exactly 9.192 631 770 GHz (the famous clock transition) between these ground states is much less than the optical frequency of $c/(894.6\,\text{nm}) \simeq 335$ THz between the ground and excited states, the resonance condition for this two-photon transition is defined by the difference between the frequencies of the two lasers. Consequently, only the difference frequency which is in the much more easily accessible microwave regime must be precisely controlled. In order to transfer from the $F=3$ state to the $F=4$ state, an atom must undergo a stimulated *absorption* process from the first laser and a stimulated *emission* process into the second laser. If the two lasers always counterpropagate, the atom recoils in the same direction during both processes. Thus, as an additional benefit, by driving two-photon transitions, we double the size of the recoil and quadruple the size of the recoil frequency shift ω_{rec}.

The goal is to transfer atoms efficiently between ground states (from $|a\rangle$ to $|b\rangle$ and back) without disturbing their relative phase. At the atomic densities and temperatures used in our experiment, state changing collisions are extremely rare. Thus, for laboratory time scales, the hyperfine ground states live essentially forever. The excited states, on the other hand, are stable for only roughly 30 ns before they spontaneously decay back into one of the ground states. The spontaneous decay process is driven by coupling to the background vacuum field. Because the number of these vacuum states is infinite, the chance that any two spontaneous decay processes will result in the same final phase is vanishingly small. Thus, each spontaneous decay process produces a random final phase value. Since the lifetime of the excited states is much smaller than the millisecond time scales of our interferometers, the randomizing spontaneous decay process immediately destroys any coherence. Therefore, we must avoid any coupling to the excited levels. There are two general methods for transferring atoms between states $|a\rangle$ to $|b\rangle$ while at the same time minimizing the coupling to the decohering excited levels. They are *off-resonant Raman* and *dark-state adiabatic passage*. Off-resonant Raman techniques have been successfully used in atom interferometry [14, 15]. In this technique both laser frequencies are detuned by many excited state natural linewidths ($\Delta \sim 200\Gamma$) from single-photon resonance, thereby greatly suppressing the coupling to the excited states but still driving two-photon transitions. In this case the effective two-photon Rabi frequency will be $\Omega_{\text{eff}} = \Omega_1\Omega_2/\Delta$. To generate a pulse which transfers atoms from one ground state to another, one simply fixes the laser frequencies and exposes the atoms to the laser light for a time τ such that $\theta = \int_0^\tau \mathrm{d}t\, \Omega_{\text{eff}}(t) = \pi$. The case when $\theta = \pi$ is called a π-pulse. A $\pi/2$-pulse is created when the integrated effective Rabi frequency $\theta = \pi/2$. In this case, an atom starting in one of the pure states ($|a\rangle$ or $|b\rangle$) ends in a quantum superposition of states $|a\rangle$ and $|b\rangle$.

For the dark-state adiabatic transfer technique which is used in this experiment [24, 25], both lasers are tuned exactly on resonance ($\Delta = 0$). To see how this technique avoids coupling to the excited states, we first write the Hamiltonian for the atom plus laser system in the $\{|\mathrm{a}\rangle\langle\mathrm{a}|, |\mathrm{i}\rangle\langle\mathrm{i}|, |\mathrm{b}\rangle\langle\mathrm{b}|\}$ pure-state basis.

$$H = -\frac{\hbar}{2}\begin{bmatrix} 0 & \Omega_1 e^{-i[\phi_1+(\Delta-\delta/2)t]} & 0 \\ \Omega_1 e^{+i[\phi_1+(\Delta-\delta/2)t]} & i\,\Gamma & \Omega_2 e^{+i[\phi_2+(\Delta+\delta/2)t]} \\ 0 & \Omega_2 e^{-i[\phi_2+(\Delta+\delta/2)t]} & 0 \end{bmatrix} \tag{2.70}$$

Diagonalizing this Hamiltonian produces a new basis of three states: $|\mathrm{BS1}(t)\rangle$, $|\mathrm{BS2}(t)\rangle$, and $|\mathrm{DS}(t)\rangle$. The first two states are "bright states" because they still include the excited state $|\mathrm{i}\rangle$. The last state can be written in the original pure-state basis

$$\begin{aligned} |\mathrm{DS(t)}\rangle &= \begin{bmatrix} c_1(t) & c_2(t) & c_3(t) \end{bmatrix}\begin{bmatrix} |\mathrm{a}\rangle \\ |\mathrm{i}\rangle \\ |\mathrm{b}\rangle \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta(t)\ |\mathrm{a}\rangle \\ 0\ \ |\mathrm{i}\rangle \\ -\sin\theta(t) e^{-i[\delta t+\phi_2-\phi_1]}\ |\mathrm{b}\rangle \end{bmatrix} \end{aligned} \tag{2.71}$$

where

$$\tan\theta(t) = \Omega_1(t)/\Omega_2(t) \tag{2.72}$$

Note that this "dark state" does not include the excited state $|\mathrm{i}\rangle$ and thus will remain coherent. To understand how such a dark state can arise when both light fields are on resonance, consider the case when the atom is in state $|\mathrm{a}\rangle$. If the first laser were off ($\Omega_1 = 0$), then $\theta = \tan^{-1}(0/\Omega_2) = \tan^{-1}(0) = 0$, and $|\mathrm{DS}\rangle = \{1, 0, 0\} = |\mathrm{a}\rangle$. The second laser couples resonantly only to state $|\mathrm{b}\rangle$ so an atom in state $|\mathrm{a}\rangle$ will not be driven to the excited state, and thus it is in the dark state. More generally, consider arbitrary laser strengths Ω_1 and Ω_2 interacting with some state $|\psi\rangle = c_1|\mathrm{a}\rangle + c_3|\mathrm{b}\rangle$. The coupling between this state and the excited state is $\langle\mathrm{i}|H|\psi\rangle = c_1\Omega_1 e^{+i[\phi_1+(\Delta-\delta/2)t]} + c_3\Omega_2 e^{+i[\phi_2+(\Delta+\delta/2)t]}$. By appropriately choosing c_1 and c_3 as in equation (2.71), the excited state coupling with the fraction of $|\psi\rangle$ in the state $|\mathrm{a}\rangle$ exactly cancels the excited state coupling with the fraction of $|\psi\rangle$ in state $|\mathrm{b}\rangle$: $\langle\mathrm{i}|H|\psi\rangle = 0$.

To actually transfer the atoms from one state to another, the relative intensities of the two laser fields are varied in time. At any given instant, there will be a well-defined dark state. As the relative laser intensities (or frequencies) change, this dark state will also change. However, if the change is slow enough, atoms starting in the dark state will remain there and adiabatically follow the changing state. The time scale which roughly defines the rate of change for which the process will still be adiabatic is the effective Rabi frequency, Ω_{eff}. For on-resonance dark-state transfer,

$$\Omega_{\text{eff}} = \sqrt{\Omega_1^2 + \Omega_2^2} \tag{2.73}$$

If τ is the time scale of some change of the dark state, then when $\Omega_{\text{eff}}\,\tau \gg 1$, the process will be adiabatic. In the other regime when $\Omega_{\text{eff}}\tau \ll 1$, the change of the dark state is too fast for the atoms to follow. In the limit of an infinitely fast change, the atomic state will simply be projected onto the new basis defined by the light fields after the change. The part of the atomic state projecting onto either of the two bright states will couple to the excited states and absorb a single photon. Once in the excited state, the atom almost immediately falls back into one of the ground states with randomized phase. There is a chance that it will fall back into the dark state (see Section 2.3.1). If it falls back into the bright state, it will once again absorb and then spontaneously emit a photon. This process continues until the atom ends up either 1) in the dark state, 2) optically pumped to a magnetic sublevel that has no excited state with which the light fields can couple, or 3) pushed via optical recoils out of the spatial extent of the laser beams. The remainder of the atomic state will be in the dark state with its phase preserved.

To construct a pulse similar to the π-pulse of the off-resonant Raman technique, we apply a pulse of light whose light intensities vary as shown in Figure 2.7. In this case, the atoms begin in state $|\text{a}\rangle$. At the beginning of the pulse, the Ω_1 light is off and Ω_2 turns on rapidly. With just Ω_2 on, the dark state is $|\text{a}\rangle$, so the overlap between the dark state and the atomic state is perfect and all of the atoms start in the dark state. After Ω_2 is fully on, we begin slowly turning on Ω_1. As Ω_1 turns on, we also begin slowly turning Ω_2 off. With both light fields present, the dark state is

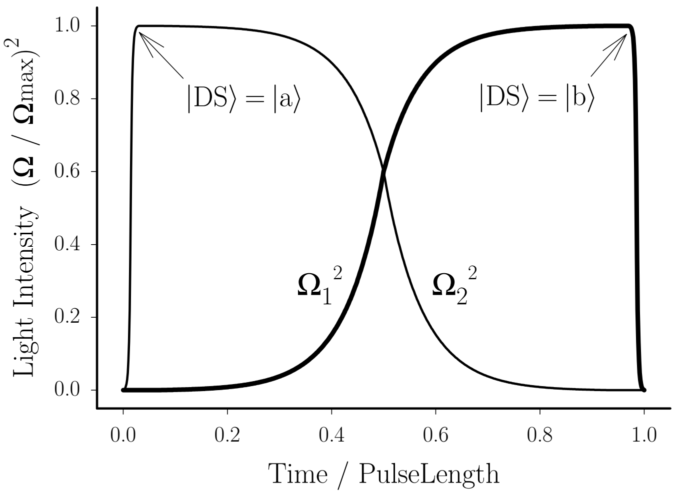


Figure 2.7: **The two laser intensities for an adiabatic passage π-pulse** which transfers atoms from state $|a\rangle$ to $|b\rangle$. The bold curve shows the intensity Ω_1^2 of the laser coupling state $|a\rangle$ to $|i\rangle$, while the other curve shows the intensity Ω_2^2 of the second laser coupling $|b\rangle$ to $|i\rangle$. For given intensities, the laser-atom coupling defines a dark state $|DS\rangle$ which is not coupled to the excited state and which changes in time as the intensities change. If the intensities change slowly enough, an atom starting in the dark state (in this case, $|a\rangle$) will remain there and evolve into the final state (in this case, $|b\rangle$) without ever coupling to the excited state.

a combination of $|a\rangle$ and $|b\rangle$. As the light intensities change, the dark state changes, but if the intensities change slowly enough, the atoms will adiabatically follow this change and stay in the dark state. At the middle of the pulse when the light intensities are equal, the atom is in an equal superposition of $|a\rangle$ and $|b\rangle$. If at this point, both light fields were shut off rapidly as shown in Figure 2.8a, the atoms would stay in this superposition state, similar to a $\pi/2$-pulse. For a π-pulse, the light intensities continue slowly changing until at the end of the pulse only Ω_1 is on and the dark state is $|b\rangle$. Finally, Ω_1 is turned off suddenly and the atoms are projected onto the original pure state basis. Since the atoms are all in $|b\rangle$, the overlap is again perfect and all of the atoms remain in state $|b\rangle$.

It is important to note that although this dark-state transfer process is analogous to the π-pulse of off-resonant Raman transfer, it is not identical. An off-resonant Raman π-pulse will transfer atoms to the other hyperfine ground state independent of their initial state: $|a\rangle$ goes to $|b\rangle$ and $|b\rangle$ to $|a\rangle$. For dark-state adiabatic transfer, however, because the light fields always determine the state of the atoms, the pulse

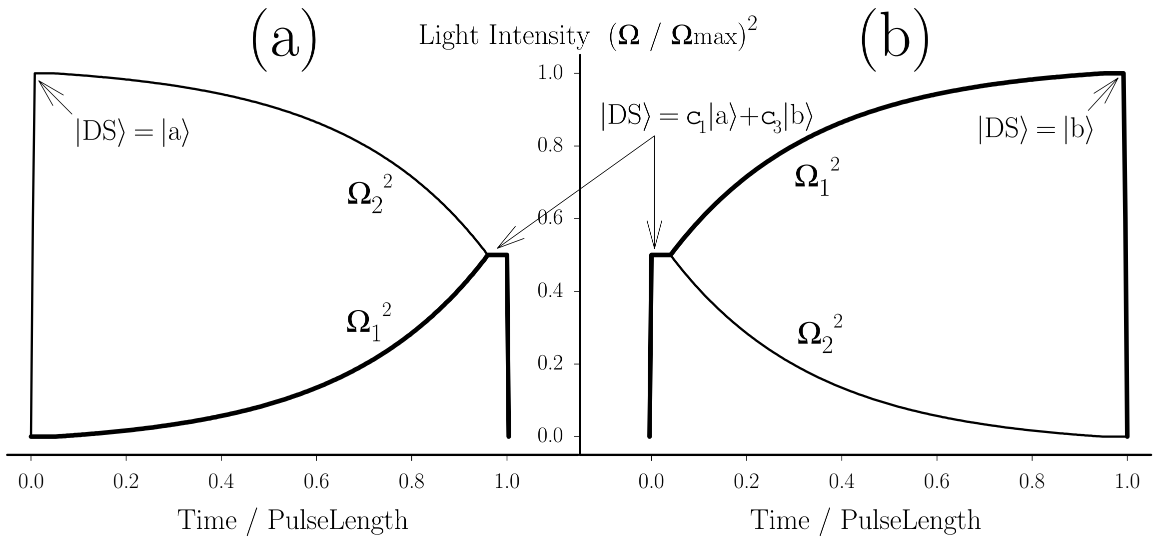


Figure 2.8: **The two laser intensities for two adiabatic passage $\pi/2$-pulses.** The bold curves show the intensity Ω_1^2 of the laser coupling state $|a\rangle$ to $|i\rangle$, while the other curves show the intensity Ω_2^2 of the second laser coupling $|b\rangle$ to $|i\rangle$. The pulse shape in (a) transfers the atoms from state $|a\rangle$ to an equal superposition of states $|a\rangle$ and $|b\rangle$, while the $\pi/2$-pulse in (b) transfers the atoms from an equal superposition state to $|b\rangle$. At the beginning and end of the pulses, the light intensities change rapidly. This non-adiabatic or sudden change in the dark state is too fast for the atoms to follow. At the beginning of the pulses, the atomic state is projected onto the dark state $|DS\rangle$ defined by the light. At the end of the pulses, the atomic state projects onto the pure state basis $\{|a\rangle\langle a|, |i\rangle\langle i|, |b\rangle\langle b|\}$.

will only transfer atoms from one state to the other but not the other way around. In the above example, for instance, the "AB-pulse" shape transfers atoms only from $|a\rangle$ to $|b\rangle$. Any atoms in $|b\rangle$ at the beginning of the pulse are in the bright state and will be driven to the excited state where they will spontaneously emit a decohering photon. This dark-state transfer pulse is therefore not a true π-pulse. However, because of its convenience and historical significance, throughout this work, we will refer to these general pulse shapes as π- and $\pi/2$-pulses.

Building the remaining pulse shapes is a simple matter of time reversing the one shape or reversing the role of the Ω_1 and Ω_2. For instance, reversing the roles of Ω_1 and Ω_2 in Figure 2.7 constructs a BA-pulse which transfers atoms from $|b\rangle$ to $|a\rangle$. The pulse shape in Figure 2.8a is an AS-pulse because it transfers atoms from $|a\rangle$ to a superposition state. Time reversing this pulse constructs an SA-pulse. Interchanging the roles of Ω_1 and Ω_2 construct the remaining BS- and SB-pulses. An SB-pulse is shown in Figure 2.8b.

The primary advantage of dark-state transfer is its high transfer efficiency. In similar experiments using the off-resonant Raman technique, π-pulse efficiencies are typically on the order of 90% [26]. In our experiment, dark-state adiabatic passage transfers atoms from one ground state to the other with around $(1-\epsilon) \sim 94\%$ efficiency. Theoretically, the maximum transfer efficiency is around 99%, limited by the presence of the of the other hyperfine excited state. Largely for this reason, we use the "D1"-transition of cesium at 894.6 nm where the $6P_{1/2}$ excited state hyperfine splitting is 1.17 GHz instead of the "D2"-transition at 852.4 nm whose excited state $6P_{3/2}$ has a much smaller hyperfine splitting of roughly 0.2 GHz (see Figure 2.6). Besides the fundamental limit set by the presence of the other excited state, the transfer efficiency is also limited in practice by pulse shapes that do not change perfectly adiabatically. For a given finite laser power, we are always limited in how long we can generate the pulse, because as the pulse duration gets longer, its frequency width becomes narrower and narrower until it becomes less than the Doppler width of the atomic cloud. At this point, many of the atoms are Doppler shifted out of two photon resonance. The light does not drive these atoms as strongly and thus they are less able to follow the change of the dark state. In addition to the effect of the narrowing linewidth, long π-pulses also suffer because as gravity accelerates the atoms during the pulse, they Doppler shift out of resonance. Although we change the laser difference frequency between pulses to compensate for the Doppler shift due to gravity, we do not linearly sweep the frequency during the π-pulses. This makes the beginning and end of the pulses less adiabatic. However, even with these practical limitations, we are still able to achieve much better efficiency using adiabatic dark-state transfer than using only off-resonant Raman transfer.

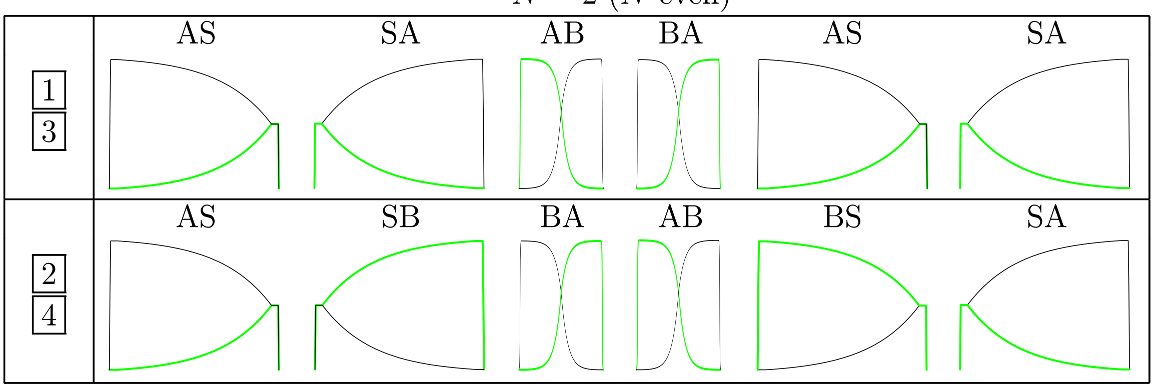
2.3 Interferometers using adiabatic transfer

To construct all of the various interferometer geometries discussed in Sections 2.1.1 through 2.1.4, we must assemble the appropriate adiabatic transfer pulse shapes. For interferometer geometry $\boxed{1}$ with no π-pulses we require four $\pi/2$-pulses with shape

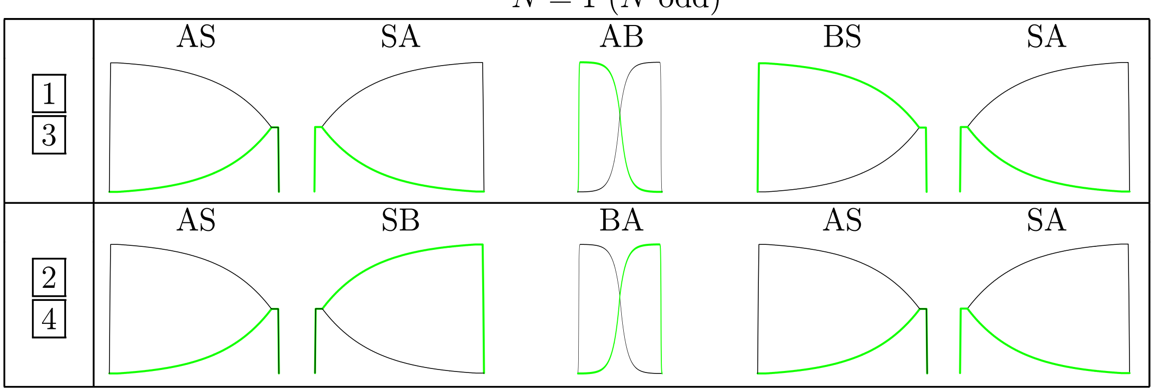
AS, SA, AS, SA, where AS indicates a $\pi/2$-pulse that transfers atoms from $|\mathrm{a}\rangle$ to the superposition state with roughly equal parts of $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$. For the conjugate interferometer $\boxed{2}$, the atom must be in the other hyperfine ground state between the second and third $\pi/2$-pulses. Building this interferometer thus requires pulse shapes AS, SB, BS, SA. For our interferometers, the atoms always start and end in the $|\mathrm{a}\rangle$, so only the middle two $\pi/2$-pulses ever change shape. Interferometer $\boxed{3}$ is the same as $\boxed{1}$ with all of the recoil directions reversed, so it is constructed with the same pulse shapes, and similarly for interferometers $\boxed{2}$ and $\boxed{4}$. If the polarity of N changes, then the third $\pi/2$-pulse must once again interchange between AS and BS pulses. Table 2.3 summarizes the pulse shapes required to build each interferometer.

Table 2.3: **Laser intensity pulse shapes** required to generate the four interferometer geometries $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, and $\boxed{4}$. The gray curves show the intensity of the laser coupling state $|\mathrm{a}\rangle$ to $|\mathrm{i}\rangle$, while the other curve shows the intensity of the second laser coupling $|\mathrm{b}\rangle$ to $|\mathrm{i}\rangle$. Each of the these geometries incorporate four $\pi/2$-pulses and N additional π-pulses inserted between the second and third $\pi/2$-pulses. Here A, B, and S represent states $|\mathrm{a}\rangle$, $|\mathrm{b}\rangle$, and the equal superposition state, respectively. Thus, an AS pulse shape transfers the atoms from state $|\mathrm{a}\rangle$ to an equal superposition of states $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$. The only difference between interferometer $\boxed{1}$ and $\boxed{2}$ is the atomic state after the second $\pi/2$-pulse, and similarly for the inverted interferometers $\boxed{3}$ and $\boxed{4}$. When N is odd, the atomic state at the beginning of the third $\pi/2$-pulse is reversed. The initial atomic state entering and the final atomic state leaving the interferometers is arbitrary. In our case, we always begin and end in state $|\mathrm{a}\rangle$, so the first and fourth $\pi/2$-pulse shapes are always the same for the different interferometer geometries.

$N = 2$ (N even)



$N = 1$ (N odd)



Ignoring imperfections in the pulses, the final interferometer phase difference does not depend on whether the interferometer is constructed using off-resonant Raman or adiabatic transfer pulses. However, for adiabatic dark-state transfer there is an important distinction for the second and fourth $\pi/2$-pulses. Just before the light for these pulses turns on, the atoms are in a coherent superposition state, $|\psi_-\rangle = (|\mathrm{a}\rangle + e^{-i\phi_\mathsf{A}}|\mathrm{b}\rangle)/\sqrt{2}$. This single quantum state is spread over two distinct spatial paths, where ϕ_A represents the total phase difference that has evolved between the two paths. Both light fields Ω_1 and Ω_2 turn on at the same time to the same level with a well defined phase difference ϕ_L. These light fields define a dark state given by equation (2.71), $|\mathrm{DS}\rangle = (|\mathrm{a}\rangle - e^{-i\phi_\mathsf{L}}|\mathrm{b}\rangle)/\sqrt{2}$. Instead of causing the atomic state to start Rabi oscillations as an off-resonant Raman process would, these adiabatic transfer pulses project the atomic state onto the dark state and then adiabatically change the dark state to end up with a pure state. Because the atoms have been evolving for time T in a superposition of two states whose velocity differ by one recoil v_r, the two paths of the interferometer will be spatially distinct as long as $v_\mathrm{r}T > \Delta r$, where Δr is the spatial spread of the individual atomic wavefunctions. Even for Δr as large as one wavelength, all reasonable experimental values for T satisfy this relation. When the light fields for the second $\pi/2$-pulse turn on, they thus interact separately with the two parts of the atomic wavefunction, $|\psi_-\rangle(\text{shifted}) = |\psi_{\mathrm{S}-}\rangle = -e^{-i\phi_\mathsf{A}}|\mathrm{b}\rangle$ for the shifted path and $|\psi_-\rangle(\text{unshifted}) = |\psi_{\mathrm{U}-}\rangle = |\mathrm{a}\rangle$ for the unshifted path. After the projection, these atomic states become

$$\begin{aligned}|\psi_{\mathrm{S}+}\rangle = |\mathrm{DS}\rangle\langle\mathrm{DS}||\psi_{\mathrm{S}-}\rangle &= |\mathrm{DS}\rangle\frac{1}{2}(\langle\mathrm{a}| - e^{+i\phi_\mathsf{L}}\langle\mathrm{b}|)(-e^{-i\phi_\mathsf{A}}|\mathrm{b}\rangle) \\ &= -\frac{1}{2}\exp(i(\phi_\mathrm{L} - \phi_\mathrm{A})))|\mathrm{DS}\rangle \end{aligned} \tag{2.74}$$

and

$$\begin{aligned}|\psi_{\mathrm{U}+}\rangle = |\mathrm{DS}\rangle\langle\mathrm{DS}||\psi_{\mathrm{U}-}\rangle &= |\mathrm{DS}\rangle\frac{1}{2}(\langle\mathrm{a}| - e^{+i\phi_\mathsf{L}}\langle\mathrm{b}|)|\mathrm{a}\rangle \\ &= \frac{1}{2}|\mathrm{DS}\rangle \end{aligned} \tag{2.75}$$

both of which then follow the dark state as it adiabatically evolves into a pure state.

No interference has taken place. The relative phase between the interferometer paths is still preserved. However, because the separate parts of the atomic wavefunction each overlap with half of the equal superposition state, the probability of finding the atom in either path is 1/4 instead of 1/2. Half of the atomic wavefunction projected into the bright state, absorbed a single photon, and then spontaneously emitted a decohering photon. What we have neglected to quantify up until this point is that each time the atom spontaneously decays from the excited state, there is a non-zero probability that it will fall back into the dark state. Because the phase of this atom is randomized by the spontaneous emission process, it does not change the final interferometer phase. However, because it is in the dark state, it will be carried along though the interferometer and ultimately detected with the final signal, thus reducing the interferometer contrast.

The third $\pi/2$-pulse, just like the first, transfers each interferometer path from a pure state to an equal superposition of states $|\mathrm{a}\rangle$ and $|\mathrm{b}\rangle$. Half of these atoms never overlap and thus never interfere. The two interferometer paths that overlap at the fourth $\pi/2$-pulse are combined together into a single wavefunction. Therefore, the overlap between the atomic state $|\psi_-\rangle$ and the dark state $|\mathrm{DS}\rangle$ it is projected onto is not limited to 1/2. Finally, the atom will emerge in the pure state $|\mathrm{a}\rangle$ with amplitude $\frac{1}{2\sqrt{2}}(1+\exp(i\Delta\phi))$, where $\Delta\phi$ is the final interferometer phase difference. The probability of finding the atom in this state is thus

$$\begin{aligned} |\frac{1}{2\sqrt{2}}(1+\exp(i\Delta\phi)|^2 &= \frac{1}{8}(1+\exp(-i\Delta\phi))(1+\exp(i\Delta\phi)) \\ &= \frac{1}{4}(1+\cos\Delta\phi) \end{aligned} \quad (2.76)$$

The phase $\Delta\phi$ is the quantity we measure in order to derive a value for the recoil frequency. In Section 5.1 we dicuss how this is done.

2.3.1 Contrast limit

To determine a theoretical limit for the contrast, we assume that the AS and BS $\pi/2$-pulses (the first and third) transfer a fraction r of the atoms from one hyperfine

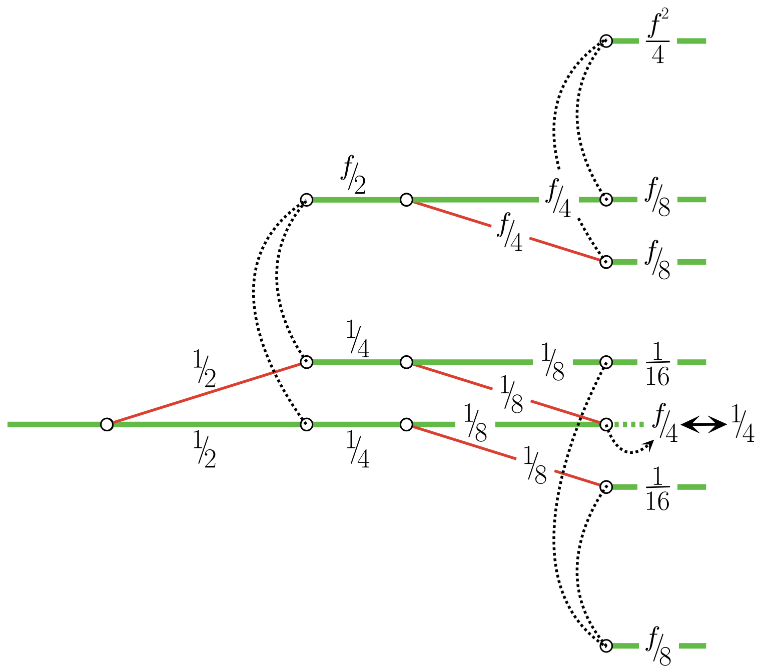


Figure 2.9: **Interference contrast limit with adiabatic passage**, assuming ideal $\pi/2$-pulses. The fraction of atoms in each path is given. At the second $\pi/2$-pulse half of the atoms project into the dark state. The other half that project into the bright state are driven to the excited state $|\mathrm{i}\rangle$. From $|\mathrm{i}\rangle$ they spontaneously emit an incoherent photon and fall back into the dark state with probability f. This process is represented by the curved dashed lines. At the last $\pi/2$-pulse, the two intersecting interferometer paths interfere. If the interference is completely constructive ($\Delta\phi = 0$), all of the atoms start the pulse in the dark state and the fraction of atoms emerging is the total fraction from both paths. On the other hand, if the interference is completely destructive ($\Delta\phi = \pi$), all of the atoms are in the bright state. As before, a fraction f of these atoms fall back into the dark state after absorbing and then emitting a single photon. Thus, the double-headed arrow at the interfering output of the interferometer represents the continuous range of possible interference phase differences $\Delta\phi = \pi \leftrightarrow 0$.

ground state to the other and leave a fraction $(1 - r)$ in the input state. Since we are interested in the theoretical maximum, for now we will assume ideal beamsplitter pulses with $r = \frac{1}{2}$. The second $\pi/2$-pulse, with shape either SA or SB projects half of the atoms onto the bright state and half of the atoms onto the dark state. The atoms in the bright state have probability f of falling back into the dark state and continuing on through the interferometer. Three distinct groups of atoms interact with the third $\pi/2$-pulse: $\frac{1}{4}$ of the atoms enter from the shifted path, $\frac{1}{4}$ from the unshifted path, and $\frac{f}{2}$ incoherent atoms scattered back into the dark state by the second $\pi/2$-pulse.

Each of these groups is again split with equal ratios. Because the atoms enter the first and third $\pi/2$-pulses in a pure state, the projection is perfect and no atoms are scattered via spontaneous emission processes back into the dark state. A fraction $\frac{1}{8}$ from each interferometer path continues on to the fourth $\pi/2$-pulse and never overlaps. This total fraction of $\frac{1}{4}$ produces no interference fringes, just a constant background. Similarly, the group of $\frac{f}{2}$ atoms split into two paths of $\frac{f}{4}$ which do not interfere. Two $\frac{1}{8}$ fractions from the shifted and unshifted paths overlap and do interfere at the fourth $\pi/2$-pulse. For perfectly constructive interference, $\frac{1}{8}+\frac{1}{8}=\frac{1}{4}$ atoms emerge. If the final total phase difference is π, these paths destructively interfere and this fraction of the atoms are all in the bright state. A fraction $\frac{f}{4}$ fall back into the dark state to be detected. When the $\frac{1}{4}$ of the atoms that do not overlap encounter the fourth $\pi/2$-pulse, half of them are projected onto the dark state and half are projected onto the bright state. The $\frac{1}{8}$ of the atoms projecting onto the dark state and the $\frac{f}{8}$ of the atoms that began in the bright state but fall back into the dark state combine for a total fraction of $\frac{1+f}{8}$ in the dark state to be detected. Also contributing to this signal are the $\frac{f}{2}$ atoms which are again projected by the fourth $\pi/2$-pulse leaving $\frac{f}{2}\frac{1+f}{2}=\frac{f(1+f)}{4}$ in the dark state. For this idealized interferometer as summarized in Figure 2.9, the minimum n_{min} and maximum n_{max} fraction of atoms detected are

$$n_{\mathrm{min}} = \frac{f}{4}+\frac{1+f}{8}+\frac{f(1+f)}{4} = \frac{1}{8}(2f+2f^2+3f+1) \tag{2.77}$$

$$n_{\mathrm{max}} = \frac{1}{4}+\frac{1+f}{8}+\frac{f(1+f)}{4} = \frac{1}{8}(2+2f^2+3f+1) \tag{2.78}$$

The contrast is thus

$$C=\frac{n_{\mathrm{max}}-n_{\mathrm{min}}}{n_{\mathrm{max}}+n_{\mathrm{min}}}=\frac{2-2f}{4f^2+8f+4}=\frac{1}{2}\frac{1-f}{f^2+2f+1}=\frac{1-f}{2(f+1)^2} \tag{2.79}$$

Next we calculate a value for f, the probability of falling back into the dark state after one or more single photon excitations. Once an atom has absorbed a single photon and transfers to the excited state it has probability p_{B} and p_{D} of falling back into the bright and dark states, respectively. If it falls back into the bright state, it will absorb another single photon, transfer to the excited state, and once again have a chance p_{D} of falling into the dark state. This process of absorbing and re-emitting single photons will continue until the atom falls into the dark state, at which point it will stop absorbing single photons. Thus, the net chance f of ending up in the dark state after $1, 2, 3, \ldots$ excitations is

$$f = p_{\mathrm{D}} + p_{\mathrm{B}}\, p_{\mathrm{D}} + p_{\mathrm{B}}\, p_{\mathrm{B}}\, p_{\mathrm{D}} + p_{\mathrm{B}}\, p_{\mathrm{B}}\, p_{\mathrm{B}}\, p_{\mathrm{D}} + \cdots \tag{2.80}$$

Since there are only three possible states, we must have $p_{\mathrm{B}} = p_{\mathrm{D}} = p$, so equation (2.80) becomes

$$f = p + p^2 + p^3 + \cdots = \frac{p}{1-p} \tag{2.81}$$

Because the dark state is a combination of particular magnetic sublevels of the cesium, the probability p depends on the branching ratios for all of the possible states accessible from the $m_F = +1$ excited state. The transition strengths for cesium are given in Table A.2 of Appendix A.1.2. For the $F = 3', m_F = +1'$ and $F = 4', m_F = +1'$ excited states the probabilities p_{3^0} and p_{4^0} of falling into the $F = 3, m_F = 0$ or $F = 4, m_F = 0$ state are

$$p_{3^0} = \frac{1}{2}\,\frac{6+6}{6+15+15+6+1+5} = \frac{1}{8} \tag{2.82}$$

$$p_{4^0} = \frac{1}{2}\,\frac{10+10}{10+1+9+10+15+3} = \frac{5}{24} \tag{2.83}$$

Inserting these two probabilities for the two possible excited states into equation (2.81) gives

$$f_{3^0} = \frac{p_{3^0}}{1-p_{3^0}} = \frac{1/8}{1-1/8} = \frac{1}{7} = 0.143 \tag{2.84}$$

$$f_{4^0} = \frac{p_{4^0}}{1-p_{4^0}} = \frac{5/24}{1-5/24} = \frac{5}{19} = 0.263 \tag{2.85}$$

Finally, using these two values in equation (2.79) give the maximum possible interferometer contrast for each of the two excited states

$$C(F\!=\!3') \;=\; \frac{1-f_{3^0}}{2(f_{3^0}+1)^2} \;=\; \frac{1-{}^{1}\!/_{7}}{2({}^{1}\!/_{7}+1)^2} \;=\; 0.328 \tag{2.86}$$

$$C(F\!=\!4') \;=\; \frac{1-f_{4^0}}{2(f_{4^0}+1)^2} \;=\; \frac{1-{}^{5}\!/_{19}}{2({}^{5}\!/_{19}+1)^2} \;=\; 0.231 \tag{2.87}$$

Because this theoretical maximum contrast is lower for the $F\!=\!4'$ excited state, we instead tune our lasers to the $F\!=\!3'$ excited state.

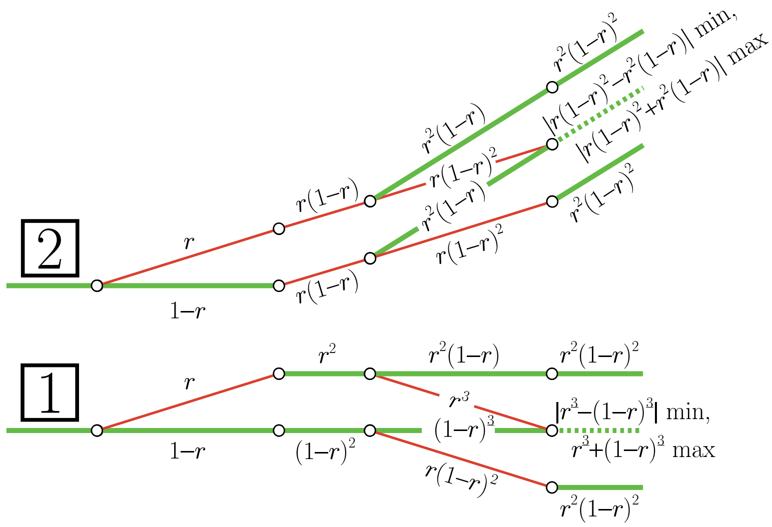


Figure 2.10: **Interference contrast limit due to imperfect $\pi/2$-pulses.** Each $\pi/2$-pulse transfers a fraction r of the input atoms into the other state and leaves alone the remaining $1-r$ atoms. At the last $\pi/2$-pulse, the two intersecting interferometer paths interfere. If the interference is completely constructive ($\Delta\phi = 0$), the fraction of atoms emerging is the total fraction from both paths, labeled "max". On the other hand, if the interference is completely destructive ($\Delta\phi = \pi$), twice the minimum fraction of the two paths is subtracted from the total fraction, labeled "min".

Practical limitations for the contrast include pulse shapes that are not perfectly adiabatic. Assuming that each $\pi/2$-pulses transfers a fraction r of the atoms into the other ground state and leaves $1-r$ in the input state. Figure 2.10 depicts how these fractions propagate through the interferometer. After the first $\pi/2$-pulse, r of the atoms are in the shifted path and the remaining $1-r$ atoms are unshifted. After the second $\pi/2$-pulse, the shifted and unshifted paths contain a fraction r^2

and $(1-r)^2$, respectively. The third $\pi/2$-pulse, splits these two groups into four groups, two of which go on to overlap at the fourth $\pi/2$-pulse and two of which never interfere. After the fourth $\pi/2$-pulse, the groups which did not interfere contribute a total fraction $2r^2(1-r)^2$ atoms to the final signal. The two paths with fractions r^3 and $(1-r)^3$ interfere at the fourth $\pi/2$-pulse either constructively, producing a total fraction $r^3 + (1-r)^3$, or destructively, producing a fraction of only $|r^3 - (1-r)^3|$. We thus have

$$\begin{aligned} n_{\text{min}} &= |r^3 - (1-r)^3| + 2r^2(1-r)^2 && (2.88) \\ n_{\text{max}} &= r^3 + (1-r)^3 + 2r^2(1-r)^2 && (2.89) \end{aligned}$$

To arrive at a more tractable expression for the contrast we now assume that like the π-pulses, the $\pi/2$-pulses transfer the atoms from one ground state to the other with efficiency of $1-\epsilon$. Thus, $r = \frac{1}{2}(1-\epsilon)$ and $(1-r) = \frac{1}{2}(1+\epsilon)$ and in the limit that $\epsilon \ll 1$ we have

$$\begin{aligned} n_{\text{min}} &\simeq \frac{1}{8}(1+3\epsilon) - \frac{1}{8}(1-3\epsilon) + 2\frac{1}{4}(1-2\epsilon)\frac{1}{4}(1+2\epsilon) \\ &\simeq \frac{1}{8}(3\epsilon + 3\epsilon + 1) \;=\; \frac{1}{8}(1+6\epsilon) && (2.90) \end{aligned}$$

$$\begin{aligned} n_{\text{max}} &\simeq \frac{1}{8}(1+3\epsilon) + \frac{1}{8}(1-3\epsilon) + 2\frac{1}{4}(1-2\epsilon)\frac{1}{4}(1+2\epsilon) \\ &\simeq \frac{1}{8}(2+1) \;=\; \frac{3}{8} && (2.91) \end{aligned}$$

producing a contrast of

$$C = \frac{n_{\text{max}} - n_{\text{min}}}{n_{\text{max}} + n_{\text{min}}} = \frac{3 - 1 - 6\epsilon}{3 + 1 + 6\epsilon} \simeq \frac{2 - 6\epsilon}{4} = \frac{1}{2} - \frac{3}{2}\epsilon \qquad (2.92)$$

For a transfer efficiency of $(1-\epsilon) = 94\%$, $\epsilon = 6\%$, and the contrast is reduced by 9%. Including the effect discussed in the previous sections where atoms fall back into the dark state and assuming we use the $F\!=\!3'$ excited state, the non-unity transfer efficiency of the pulses reduces the contrast to approximately 24%.

Since the conjugate interferometer $\boxed{2}$ uses the other hyperfine ground state the two interfering fractions are $r^2(1-r)$ and $r(1-r)^2$ which produces a contrast of

$$C = \frac{3-1-2\epsilon}{3+1+2\epsilon} \simeq \frac{2-2\epsilon}{4} = \frac{1}{2} - \frac{1}{2}\epsilon \tag{2.93}$$

equivalent to a contrast reduction of $\sim$3%.

2.3.2 AC-stark shifts

Here we consider how unavoidable off-resonant couplings affect the dark state. In equations (2.66) and (2.67) we wrote the Rabi frequencies for the first laser field $\mathbf{E}_1$ coupling with the $|\mathrm{a}\rangle \to |\mathrm{i}\rangle$ transition and the second laser field $\mathbf{E}_2$ coupling with the $|\mathrm{b}\rangle \to |\mathrm{i}\rangle$ transition. There are also two additional couplings for the reversed situation when the second laser field $\mathbf{E}_2$ couples with the $|\mathrm{a}\rangle \to |\mathrm{i}\rangle$ transition and *vice versa*. Both of these couplings are detuned from resonance by the ground state hyperfine splitting ω_{ab}. Since this detuning is larger than all of the other terms in the Hamiltonian, to an excellent approximation these off-resonant couplings modify the Hamiltonian in equation (2.70) by

$$\Delta H = \hbar \begin{bmatrix} \Delta_{\mathrm{a}}^{\mathrm{AC}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\Delta_{\mathrm{b}}^{\mathrm{AC}} \end{bmatrix} \tag{2.94}$$

where

$$\Delta_{\mathrm{a}}^{\mathrm{AC}} = \frac{\Omega_2^2}{4\omega_{\mathrm{ab}}} \tag{2.95}$$

$$\Delta_{\mathrm{b}}^{\mathrm{AC}} = \frac{\Omega_1^2}{4\omega_{\mathrm{ab}}} \tag{2.96}$$

are the ac-stark shifts. As a first-order perturbation the dark state energy eigenvalue will change from zero to

$$E_{\mathrm{DS}} = \langle \mathrm{DS}|\Delta H|\mathrm{DS}\rangle$$

$$
\begin{aligned}
&= \hbar \begin{bmatrix} \cos\theta & 0 & -\sin\theta \end{bmatrix} \begin{bmatrix} \Delta_a^{AC} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\Delta_b^{AC} \end{bmatrix} \begin{bmatrix} \cos\theta \\ 0 \\ -\sin\theta \end{bmatrix} \\
&= \hbar \begin{bmatrix} \cos\theta & 0 & -\sin\theta \end{bmatrix} \begin{bmatrix} \Delta_a^{AC}\cos\theta \\ 0 \\ \Delta_b^{AC}\sin\theta \end{bmatrix} \\
&= \hbar(\Delta_a^{AC}\cos^2\theta - \Delta_b^{AC}\sin^2\theta) \\
&= \frac{\hbar}{4\omega_{ab}}\left(\frac{\Omega_2^4 - \Omega_1^4}{\Omega_{eff}^2}\right) \\
&= \frac{\hbar}{4\omega_{ab}}\left(\Omega_2^2 - \Omega_1^2\right) \qquad (2.97)
\end{aligned}
$$

where we have used equation (2.72) to evaluate θ. Similarly, to lowest order the perturbed dark state $|DS'\rangle$ can be shown to be

$$
|DS'\rangle = A \begin{bmatrix} \Omega_2(1 + i\phi_1)\ |a\rangle \\ -\frac{\Omega_1\Omega_2}{2\omega_{ab}}\ |i\,\rangle \\ -\Omega_1(1 - i\phi_2)\ |b\rangle \end{bmatrix} \qquad (2.98)
$$

where A is required for normalization and

$$
\phi_i = \frac{\Gamma}{2\omega_{ab}}\frac{\Omega_i^2}{\Omega_{eff}^2} \qquad (2.99)
$$

The dark state is thus no longer perfectly dark and is shifted in phase by

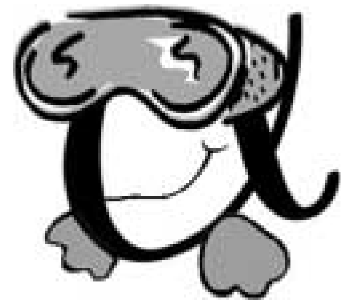
$$
-\phi_2 - \phi_1 = -\frac{\Gamma}{2\omega_{ab}}\frac{\Omega_2^2 + \Omega_1^2}{\Omega_{eff}^2} = -\frac{\Gamma}{2\omega_{ab}} \qquad (2.100)
$$

which amounts to a constant $-\Gamma/2\omega_{ab} = -4.6\,\text{MHz}/[2(9.2\,\text{GHz})] = -0.25$ mrad. We have simulated the interferometers by numerically integrating the Schrödinger equation during each of the four $\pi/2$-pulses and tracking the atom's state evolution over both interferometer paths. Even with the actual pulse shapes used in the laboratory, the simulation indicates that this ac-stark shift, although present for a single pulse, cancels after four pulses when the two interferometer paths recombine. If the pulse shapes are not the same under time reversal and exchange of Ω_1 and Ω_2, it will not completely cancel. However, if the same imbalance exists for the conjugate interferometer, the shift will subtract away with the difference between the up and down interferometers.

Another source of ac-stark shifts is the other hyperfine excited state. In this case, because of the excited state hyperfine splitting both light fields are detuned by the same amount of ± 1.17 GHz, where the sign depends on which excited state the lasers are locked to (negative for $F = 3'$). Because this ac-stark effect shifts each of the ground states used in the interferometers by the same amount, it should not cause a measurement error.

Chapter 3

Experiment



3.1 Cesium fountain

Our source of cold cesium atoms is provided by a beam-loaded magneto-optic trap (MOT) [27, 28, 29, 30, 31]. As shown in Figure 3.1, a 5 g sample of 99.98% pure cesium[1] metal is placed inside a vacuum chamber composed of a crushable can holding the cesium ampule, two bakeable all-metal valves, and a 0.6 mm diameter nozzle. This chamber, called the oven arm, is wrapped with two fiberglass heater tapes. One heater tape, called the storage heater, is wrapped around the crushable can where the cesium is located. The other tape, called the nozzle heater, is wrapped around the rest of the oven arm. To prevent reactive cesium and its byproducts from condensing in the nozzle and clogging it, the nozzle heater is always on. This holds the nozzle at around 210°C, which is always warmer than the rest of the oven arm chamber. The storage heater, on the other hand, is shut off when we are not running the experiment. This brings regions of the storage can below the sublimation point of solid cesium. Cesium condenses in these regions and does not leave the oven arm, thereby conserving cesium. When the storage heater is on, the coldest part of the storage can is around 120°C and the metal cesium sublimates into a

[1] Cesium has only one naturally occuring stable isotope, so our cesium sample consists *entirely* of ^{133}Cs.

vapor, which sprays ballistically out of the nozzle into the lower pressure region of the source chamber. From the measured temperature of the cesium can, we calculate a minimum root-mean-square (rms) velocity of $\sqrt{3k_{\rm B}T/m} \simeq 271$ m/s. From vapor pressure data for cesium [32], we can also estimate a density of $n \sim 4 \times 10^{13}\,\text{cm}^{-3}$ and thus a mean distance of $1/(n\sigma\sqrt{2}) \sim 35$ cm between Cs-Cs collisions, assuming a collisional cross-section of $\sigma \sim 500 \times 10^{-16}\,\text{cm}^2$. This distance is much larger than the dimensions of the nozzle hole, thus the nozzle system should operate in the effusive regime where the velocity profile of the escaping cesium vapor is determined solely by the length and diameter of the nozzle hole. For circular apertures the collimation factor is $(8r)/(3l)$, where r and l are the nozzle radius and length [33]. For our nozzle dimensions, $r = 0.3$ mm and $l = 6.4$ mm, the ratio of the longitudinal velocity to the transverse velocity is ~ 8.

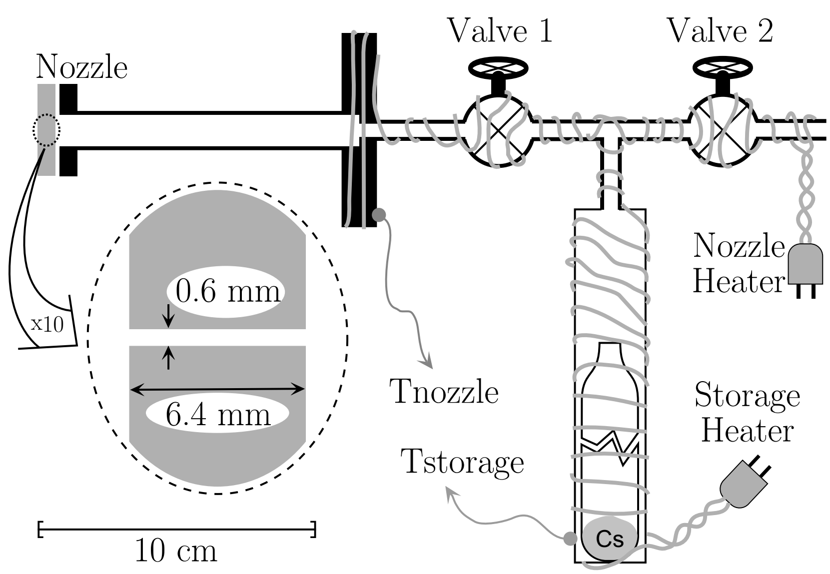


Figure 3.1: **Cesium source.** A glass ampule containing several grams of cesium metal is placed in a crushable can attached to a stainless steel vacuum chamber. After the chamber is evacuated through "Valve 2" and the ampule is broken, the chamber is heated with fiberglass insulated heater tape. As shown, two separate heater tapes are used to independently heat the "nozzle" and "storage" regions. The temperatures of these regions are measured with thermocouples located as shown. When operating, these temperatures are $T_{\text{nozzle}} = 210°\text{C}$ and $T_{\text{storage}} = 120°\text{C}$. With "Valve 1" open, gaseous cesium diffuses out through the 0.6 mm diameter circular nozzle into the larger vacuum chamber. Because the 6.4 mm length of the nozzle is much greater than the nozzle tube diameter, the escaping cesium is partially collimated. All connections are either vacuum tight welds or copper gasket sealed knife-edge flanges.

By observing the light scattered when two copropagating circularly polarized lasers tuned to the $F=4 \rightarrow F=5'$ and $F=3 \rightarrow F=4'$ D2 transitions (see Figure 2.6) intersect orthogonally with the atomic beam at the nozzle, we estimate an output flux of roughly 1×10^{13} atoms/s.

The oven arm is attached to the source chamber via vacuum-tight bellows, which allow the direction of the partially collimated atomic beam to be controlled externally. The source chamber consists of a 100 L/s turbomolecular pump and two orthogonally oriented rectangular apertures ("crossed slots") that can be moved transversely to the atomic beam with external mechanical feed-thrus. As depicted in Figure 3.2, these shutter blades along with the controllable orientation of the oven arm allow us to control the direction of the atomic beam into the main chamber. In addition to further collimating the atomic beam, the shutter blades also limit the leakage of background gases from the source chamber ($P_{\text{source}} \sim 4 \times 10^{-9}$ torr) into the main chamber, thereby allowing the main chamber to be held at an even lower pressure of around 2×10^{-9} torr. Between the source chamber and the main chamber is a 4 inch gate valve that allows us to isolate the two chambers and minimize possible contamination of the main chamber. Overlapped with the entire atomic beam but propagating in the opposite direction is a circularly polarized laser beam tuned near but below the $F=4 \rightarrow F=5'$ transition. As described below in Section 3.1.2, this "slowing beam" is frequency chirped at a rate designed to reduce the longitudinal velocity of groups of atoms emerging from the oven arm nozzle to a speed slow enough that they can be caught in the trap. To maximize the efficiency of this capture process, the slowing beam enters the chamber with a Gaussian beam diameter of 13 mm and focuses eventually to the size of approximately 2 mm at the oven arm nozzle.

Inside the main chamber, the atoms are loaded into the MOT. Inside the roughly $1\,\text{m}^3$ volume of the main chamber are the two high-field anti-Helmholtz coils for the MOT. These coils are made from 0.25 inch outer diameter copper tubing, which is electrically insulated with a loose fiberglass sheath and water cooled through the 0.19 inch diameter hollow core. Each coil is wound two turns wide by four turns deep with inner and outer diameters of roughly 6 and 9 cm. The coils are separated by 6 cm with their axes along the atomic beam direction. With a current of 16 A

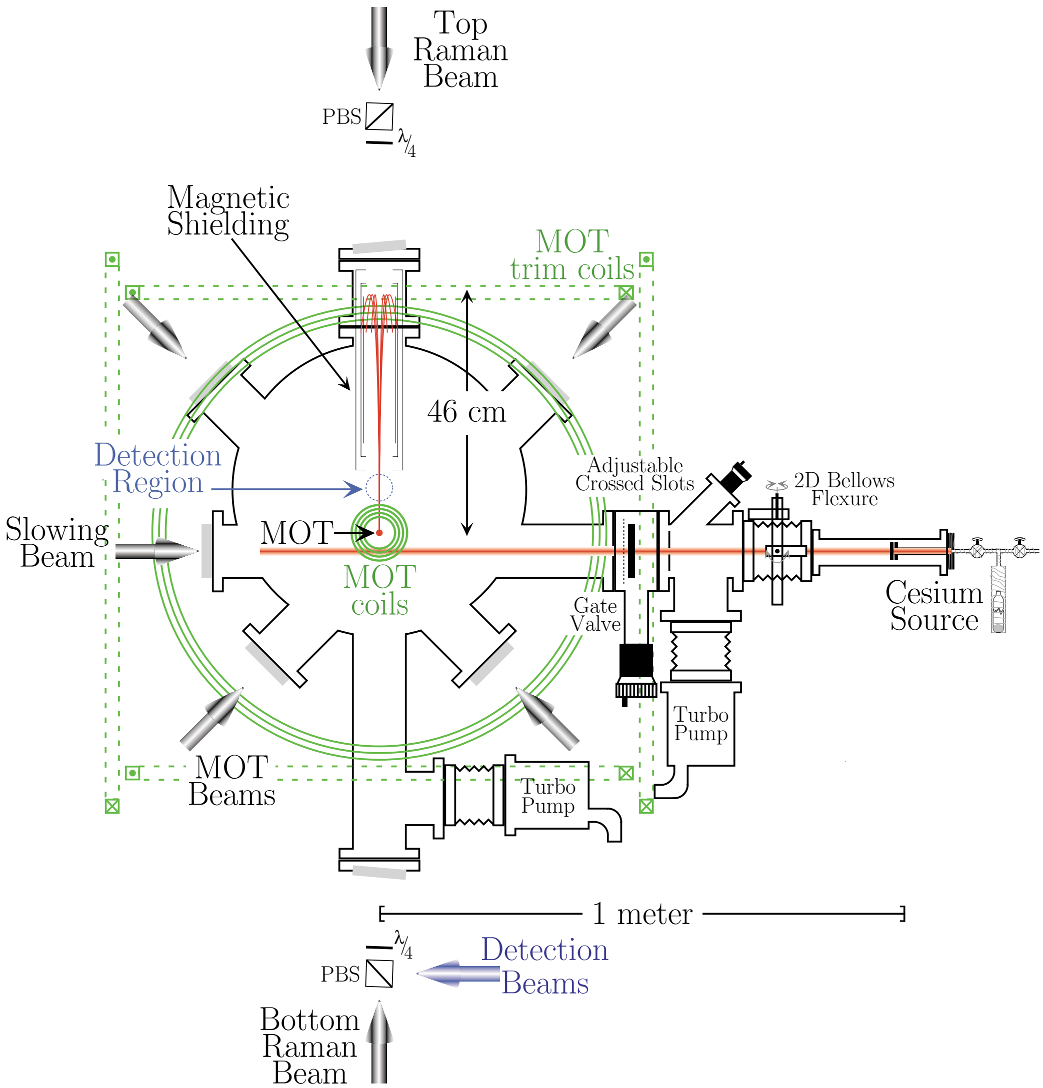


Figure 3.2: **Physical setup** shown to scale. Cesium atoms from an atomic beam are loaded at a rate of roughly 3×10^8 atoms/s into a magneto-optic trap (MOT). A frequency chirped "slowing" laser propagates against the atomic beam and slows the atoms' longitudinal velocities so that the atoms can be more efficiently trapped by the MOT. Two of the six MOT beams are not shown and propagate into and out of the page. The four beams that are shown are used to launch the atoms vertically with a velocity of $\sim$3 m/s. After traveling $\sim$46 cm upward, the atoms turn around due to gravity. At this point inside the vacuum chamber, we have installed three layers of cylindrical magnetic shielding. Centered around the apex of their trajectory, we thus have almost 400 ms with which to interact with the atoms in a perturbation free environment. It is during this time while the atoms are still within the magnetic shielding that we flash on the vertical Raman beams and build the atom interferometers. After the interference has taken place and the atoms have emerged from the shielding, we flash on a vertical "probe" beam and with a photomultiplier tube (PMT), observe the resulting fluorescence which is proportional to the number of atoms in the atomic cloud.

through the coils, this geometry provides a longitudinal and transverse magnetic field gradient of roughly 6.2 and 3.1 G/cm, respectively.

As shown in Figure 3.2, four of the six MOT beams enter the main chamber through anti-reflection (AR) coated windows in a cross configuration. The remaining two MOT beams enter normal to the plane defined by the four cross beams. All of the six MOT beams are collimated with a Gaussian waist diameter of 2.0 cm and set to the appropriate circular polarizations with 2 inch zero-order $\lambda/4$-plates before each window.

3.1.1 Laser source

Except for a small amount of repumping light tuned to the $F=3 \rightarrow F=4'$ transition, a single titanium-sapphire ring laser, assembled from a kit from the *Schwarz Electric Optics Company (SEO)*, provides all of the light necessary to trap, cool, launch, magnetically polarize, and detect the cesium atoms. When pumped with 7.4 W of light from a large frame argon-ion plasma-tube laser made by *Coherent*, this laser produces 1.0W of light at 852.3 nm. Figure 3.3 shows the fundamental components inside this laser, including two stages of active frequency stabilization.

As depicted in Figure 3.4 some of the light output is directed into an external glass cell containing cesium vapor. Using a standard technique [34, 35], we observe the dispersive component of the direct saturation absorption signal from this gas cell and use it to lock the laser output frequency to the cesium D2 transition. The saturation absorption signal produces dispersive features as the laser frequency crosses each of the allowed D2 transitions originating from the $F=4$ ground state. Because the "pump" and "probe" beams counter-propagate through the cesium cell, the width of these features is roughly 5 MHz, the natural linewidth of the $6P_{3/2}$ state, and is not limited by the much larger Doppler broadening of the room temperature cesium gas. Relative to the $F=4 \rightarrow 5'$ transition, the dispersive features occur at frequencies $f_{3^0} \simeq -450$ MHz, $f_{4^0} \simeq -250$ MHz, and $f_{5^0} = 0$. In addition to these three features, there are an additional three dispersive features at the midpoints between each pairwise combination of these transitions. Again relative to the $F=4 \rightarrow 5'$

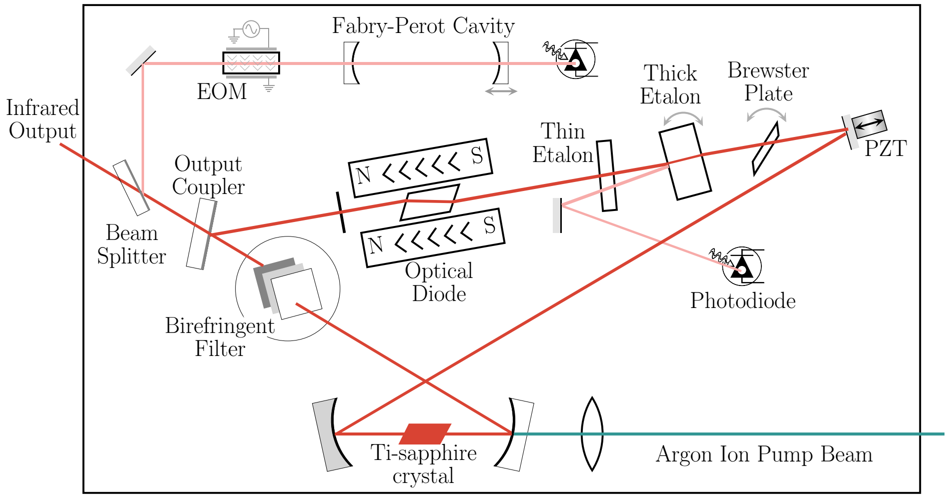


Figure 3.3: The **Titanium-sapphire ring laser** pumped by an argon ion input beam generates coherent light at infrared wavelengths. The figure-eight shaped ring cavity is constrained to resonate in only one direction by the optical diode. The remaining optical elements all have frequency dependent transmission efficiencies and thus determine the final cavity resonance frequency. In order of increasing frequency selectivity, they are: the birefringent filter, the thin etalon, the thick etalon, and finally the external Fabry-Perot cavity. The ring cavity is frequency locked to the Fabry-Perot cavity by actuating the piezo-electric transducer (PZT) mounted mirror and the Brewster plate. The layout shown here is particular to the *Schwarz Electric Optics Company* model used to generate the trap lasers at 852 nm. Another Ti-sapphire laser made by *Coherent* with a different layout but all of the same fundamental components produces the light for the Raman lasers at 894.6 nm.

transition, these additional features, often called "crossover resonances", are located at frequencies $f_{3^0/4^0} = (f_{3^0} + f_{4^0})/2 \simeq -350$ MHz, $f_{3^0/5^0} = (f_{3^0} + f_{5^0})/2 \simeq -225$ MHz, and $f_{4^0/5^0} = (f_{4^0} + f_{5^0})/2 \simeq -125$ MHz. Because it is the largest, we lock the laser to the $F=4'/5'$ crossover. Because the light going to the Cs lock first passes twice through an acousto-optic modulator (AOM) (see Figure 3.4), the absolute output frequency of the laser will be offset from this transition by twice the AOM frequency of $f_{\text{offsetAOM}} \simeq 108$ MHz. In this way the final output frequency

$$\begin{aligned} f_{\text{SEO}} &= f(F=4 \to 5') + f_{4^0/5^0} + 2f_{\text{offsetAOM}} \\ &\simeq f(F=4 \to 5') - 125\,\text{MHz} + 2(108\,\text{MHz}) \\ &\simeq f(F=4 \to 5') + 91\,\text{MHz} \end{aligned} \tag{3.1}$$

can be easily varied by changing $f_{\text{offsetAOM}}$.

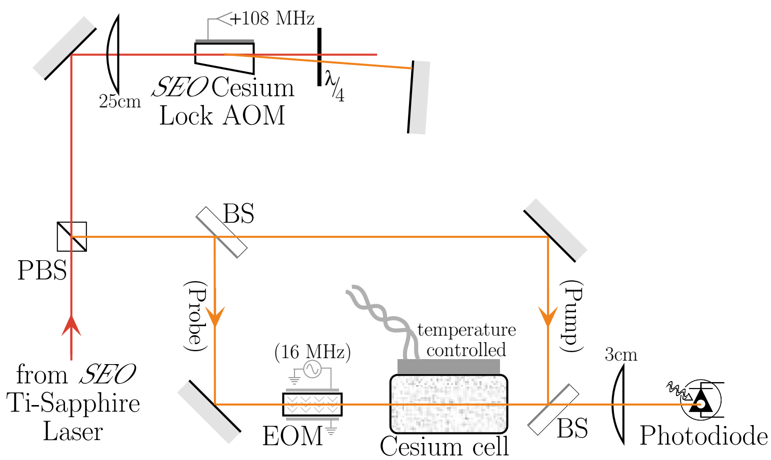


Figure 3.4: **Optical setup used to lock the *SEO* Ti-sapphire laser to the cesium transition at 852 nm.** Before interacting with the cesium atoms, light from the *SEO* Ti-sapphire laser is diffracted twice by an acousto-optic modulator (AOM). This diffracted light is then split by a dielectric beamsplitter (BS) into a "pump" and "probe" beams that counterpropagate through a temperature stabilized cell containing cesium vapor. An electro-optic modulator (EOM) phase modulates the probe beam which is observed by a photodiode. By mixing down the resulting electronic signal with a copy of the signal driving the EOM, a dispersive lock error signal can be generated. Because the pump and probe beams counterpropagate, the Doppler shifts of the individual cesium atoms cancel, and thus the width of the dispersive feature is close to natural linewidth of the atomic transition. The AOM shifts the frequency of the light by twice the frequency ($f_{\mathrm{offsetAOM}} = +108$ MHz) of its radio-frequency (rf) driving signal. Therefore, by varying $f_{\mathrm{offsetAOM}}$, one can control the laser's absolute frequency relative to the cesium transition.

3.1.2 Slowing beam

The slowing beam is generated by separating the +1 order from an *Isomet* 1250C AOM, passing it through a traveling-wave electro-optic modulator (EOM) [36], and then directing it into the vacuum chamber (see Figure 3.5). The EOM is driven by a 7.9 W rf signal that sweeps between 518 and 322 MHz in 4.9 ms. In frequency space, this phase modulation produces sidebands shifted from the carrier by the frequency of the rf signal, f_{sweep}. Thus, at most one third of the optical power emerging from the slowing EOM will have frequency

$$\begin{aligned} f_{\mathrm{slowing}} &= f_{\mathrm{SEO}} + f_{\mathrm{slowAOM}} - f_{\mathrm{sweep}} \\ &\simeq f(F{=}4 \to 5') + 91\,\mathrm{MHz} + 200\,\mathrm{MHz} - (518 \to 322\,\mathrm{MHz}) \\ &\simeq f(F{=}4 \to 5') - 227 \to -31\,\mathrm{MHz}) \end{aligned} \tag{3.2}$$

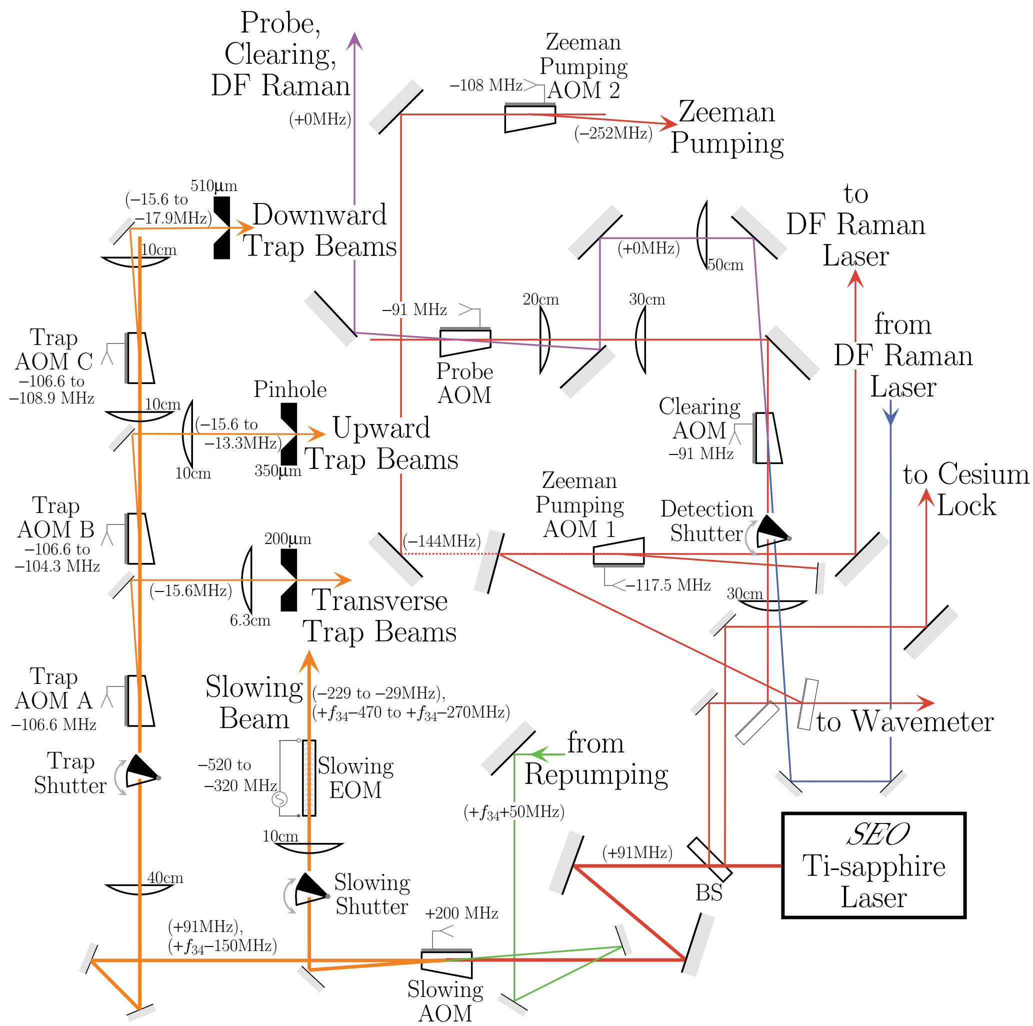


Figure 3.5: **Preparation of the laser light to cool, trap, and detect atoms in an atomic fountain.** The magneto-optic trap (MOT) beams come from a *Schwarz Electric Optics (SEO)* Ti-sapphire laser. A small fraction of this laser's output is immediately routed with a power beamsplitter (BS) to measure its wavelength with a wavemeter, lock it and the DF Raman laser diode to cesium, and generate the probe, clearing, and Zeeman pumping beams. The remaining fraction is split roughly equally by the slowing acousto-optic modulator (AOM), sending one part to generate the slowing beam and the rest to be split up into the six MOT beams. At this AOM, repumping light from a laser diode is overlapped with all of these beams. The slowing beam is frequency chirped using a traveling-wave electro-optic modulator (EOM) and then directed to the vacuum chamber. The MOT beams are generated with trap AOMs A, B, and C, which split off a fraction of the light for the transverse, upward going, and downward going trap beams, respectively. Each beam diffracted from these AOMs is focused through a pinhole to clean up its spatial profile. Note that all of the light that ultimately arrives at the vacuum chamber can be completely extinguished with mechanical shutters. The laser frequencies at various points relative to the $F=4$ to $F=5'$ D2 transition are given in parentheses ().

A linear frequency sweep of Δf in time Δt corresponds to an acceleration of $a = \lambda\Delta f/\Delta t = (852.3\,\mathrm{nm})(196\,\mathrm{MHz})/(4.9\,\mathrm{ms}) = 3.4 \times 10^4 \mathrm{m/s}^2$. Thus, due to the absorbing of photons from the slowing beam, atoms leaving the nozzle with longitudinal velocity $\sqrt{9\pi k_\mathrm{B}T/(8m)} \simeq 294$ m/s will come to rest 1.3 m later, roughly the distance from the nozzle to the center of the MOT [37].

3.1.3 MOT beams

The remainder of the light not removed by the slowing AOM, passes through the three trap AOMs, labeled A, B, and C. Each trap AOM diffracts a fraction of the light into the -1 order which separates from the main beam. The diffracted light is then focused through a pinhole, and directed into the vacuum chamber. As shown in Figure 3.5, trap AOM A generates the light for the horizontal transverse trap beam, while AOMs B and C generate light for the upward and downward going beams of the cross. To generate the two upward going beams, the light diffracted by AOM B is split by a 50/50 beam splitter so that half of the light can be sent to each side of the chamber, and similarly for the downward going beams from AOM C. The horizontal transverse beam from AOM A, on the other hand, is not split. It passes through the chamber and is then retroreflected by a mirror on the other side.

For optimal MOT performance, it is important that the atoms experience an identical magneto-optic force from each of the six trap beams. This force is proportional to the local magnetic field gradient and the local intensity of the laser. Each of the four cross beams is magnified to a collimated Gaussian beam diameter of roughly 2.0 cm. By symmetry, along each of these beam directions the magnetic field gradient is approximately the same. Therefore, since the beam size and field gradient are the same for the four cross beams, to balance the magneto-optic force, we must only control the power ratio of the these beams. The power ratio between the two upward going beams is fixed by the beam splitter, and similarly for the two downward going beams. Because the upward and downward going beam pairs come from different trap AOMs, we can continuously vary the top versus bottom power ratio by controlling the amplitude of the rf signal to the trap AOMs B and C. Along the axis of the trap coils, the gradient is approximately twice as big, and thus the intensity of the horizontal trap beams must be correspondingly smaller. The horizontal trap beam going to the chamber is magnified to a collimated Gaussian diameter of 2.0 cm. The second horizontal beam is created by retroreflecting instead of splitting the incoming beam. Because of losses through the second vacuum window and in the retro-mirror, the retroreflected beam does not have the same power as the incoming beam. To compensate, a long focal length lens is inserted after the chamber so that the beam size of the returning beam at the MOT will be slightly smaller and thus its intensity will be approximately the same.

3.1.4 Launch

Figure 3.25 shows a master timing diagram which repeats every 908 ms. The arbitrarily chosen zero occurs when the MOT coils turn off and the mechanical shutter for the slowing beam closes. For approximately 300 ms preceding this time, the MOT has been fully operational and loading atoms at a rate of around 3×10^8 atoms/s. When the slowing beam shuts off, essentially no more atoms are loaded into the trap. When the trap coils shut off, there is no longer a spatial minimum in the trapping potential. The atoms thus begin to ballistically expand in all directions. However, because the trap beams are still on, the atoms are still being Doppler cooled toward the Doppler cooling frequency limit of $\Gamma/2$ [38]. To achieve even lower temperatures we apply a small magnetic field to mostly cancel the local Earth's magnetic field and any stray fields. This field is generated by three pairs of ~ 1 m diameter coils in a Helmholtz configuration. These "MOT trim coils" consist of ~ 40 turns of 0.050 inch diameter solid copper wire and are located outside the vacuum chamber on all faces of an imaginary cube oriented normal to the atomic beam (see Figure 3.2). With zero magnetic field in the region of the trap, the atom's magnetic sub-levels are all degenerate and polarization gradient cooling takes over [39, 40, 41]. To further improve the cooling, at $t = 0$ we lower the intensities of the trap beams by a factor of ~ 150. This provides some adiabatic cooling and minimizes the heating due to the absorption and then isotropic spontaneous re-emission of photons.

Five milliseconds later at $t = 5$ ms, we change the frequencies of the cross beams relative to the horizontal beams. We shift the upward going beams up (blue shift) and the downward going beams down (red shift) by 2.3 MHz. With this frequency asymmetry, the beams effectively cool the atoms toward a non-zero velocity of 2.7 m/s in the vertical direction. This launches the atoms upward to produce the atomic fountain which serves as the source of atoms for the interferometer measurement.

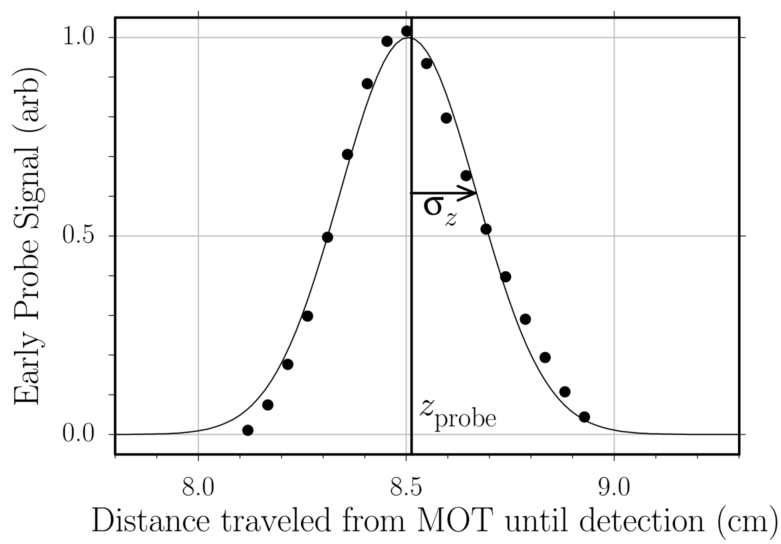


Figure 3.6: **Fluorescence from the upward traveling atom cloud.** As the atoms pass through the detection region on their way up, a vertical probe beam is flashed on for 0.3 ms. A photomultiplier tube (PMT) collects a fraction of the photons scattered from this probe beam by the atoms. For a probe pulse occuring at time t after the launch, we convert to position according to $z = v_{\mathrm{L}} t - \frac{1}{2} g\, t^2$ using a vertical launch velocity of $v_{\mathrm{L}} = 2.71$ m/s and a gravitational acceleration of $g = 9.80\,\mathrm{m/s}^2$. By varying the time of the probe pulse, we can map out the atoms' spatial distribution in the vertical direction. Fitting these data with the function $A \exp\{-\frac{1}{2}[(z - z_{\mathrm{probe}})/\sigma_{\mathrm{z}}]^2\} + B$ using the four free parameters A, z_{probe}, σ_{z}, and B gives a probe position of $z_{\mathrm{probe}} = 8.505 \pm 0.004$ cm above the MOT and an rms radius of $\sigma_{\mathrm{z}} = 1.645 \pm 0.089$ mm.

After traveling about 8 cm upward the atoms enter a region which is imaged with a reduction of ~ 3 by a 4 inch biconvex lens of focal length 8 cm onto the approximately 1×1 cm active area of a *Hammamatsu* R943-2 photomultiplier tube (PMT). When the atoms are illuminated with the vertical traveling probe beam flashed on 33.6 ms after the launch ("early probe"), they scatter light which is converted to an electrical signal by the PMT. In Figure 3.6, we present the time of flight data for the atoms as they travel upward. At this point in time, the spatial distribution of the atoms is much smaller than the size of the probe beam, so by varying the time when the probe turns on, we can extract an rms radius of 1.65 mm in the vertical dimension for the atomic cloud. Assuming the cloud is symmetric in space, this measurement indicates the size of the atomic cloud at the time of the early probe.

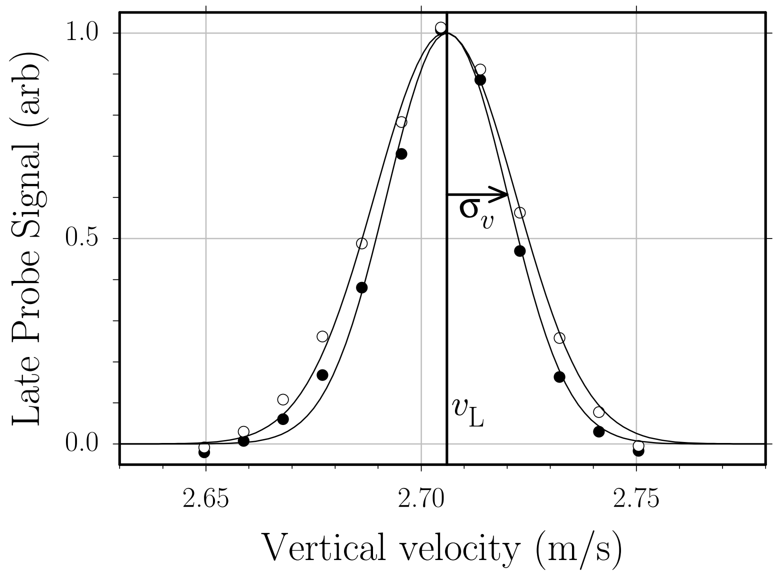


Figure 3.7: **Fluorescence from the downward traveling atom cloud.** We perform a measurement similar to the one described in Figure 3.6, but in this case we flash the probe beam on when the atoms are traveling downward instead of upward through the detection region. For a probe pulse occuring at time t after the launch, we convert to velocity according to $v = z_{\text{probe}}/t + \frac{1}{2}g\,t$ using a probe position of $z_{\text{probe}} = 8.51$ cm above the MOT and a gravitational acceleration of $g = 9.80\,\text{m/s}^2$. By varying the time of the late probe pulse, we can map out the atoms' velocity distribution in the vertical direction. The two data sets correspond to atoms launched with (hollow) and without (solid) Zeeman pumping. Fitting these two datasets with the function $A\exp\{-\frac{1}{2}[(v-v_{\text{L}})/\sigma_{\text{v}}]^2\}+B$ using the four free parameters A, v_{L}, σ_{v}, and B gives a vertical launch velocity of $v_{\text{L}} = 2.7061\pm0.0004$ m/s and an rms velocity radius of $\sigma_{\text{v}} = 1.640\pm0.058$ cm/s with Zeeman pumping and $\sigma_{\text{v}} = 1.404\pm0.050$ cm/s without.

If we perform the same measurement at 520 ms after the launch ("late probe") after the atoms have reached the top of their trajectory and returned to the detection region, we can investigate the atom's vertical velocity distribution. In order to make this measurement depend only on the velocity distribution and not on the atom's spatial distribution, we iris the probe beam from its Gaussian diameter of 12 mm down to ~ 2 mm so that it is much smaller that the falling cloud. As shown in Figure 3.7, the rms velocity spread in the vertical direction of atoms leaving the

MOT is $\sigma_v = 1.40$ cm/s, equivalent to 3.99 single photon recoils[2] or a temperature[3] of $3.18\,\mu$K. Even with this relatively cold temperature, this velocity spread produces a non-negligible spread Δz in position distribution of the atoms at the time of the early probe, $\Delta z = \sigma_v t_{\rm earlyprobe} = (14.0\,{\rm mm/s})(33.6\,{\rm ms}) = 0.472$ mm. By subtracting this spread in quadrature from the observed position spread of 1.65 mm at the time of the early probe, we can determine the rms radius of the atomic cloud at the time of the launch: $\sigma_z(t = 0) = 1.58$ mm.

3.1.5 Detection

Brief mention of the probe beam was made in the previous section. As shown in Figure 3.5, this beam is derived from the light off the first beam splitter outside the *SEO* Ti-Sapphire laser. It passes through a mechanical shutter, two AOMs (the clearing and then the probe AOM), and a final collimating lens, before it is directed into the chamber via the other input port of the bottom polarizing beam splitting (PBS) cube (see Figure 3.2). The probe beam thus propagates upward approximately overlapped with the vertical interferometer Raman beams. Overlapped with this probe beam are two other beams used for detection: the clearing and the Doppler-free (DF) Raman beam. The clearing actually comes from the same source as the probe, but it is turned on and off by the clearing AOM instead of the probe AOM. In addition, when the clearing AOM is on, the beam is deflected through a slightly different beam path which changes the beam size so that the emerging clearing beam is roughly 24 mm in diameter, or roughly twice the size of the probe beam. The probe and the clearing beams have frequency

$$\begin{aligned} f_{\rm probe} &= f_{\rm SEO} - f_{\rm probeAOM} \\ &\simeq f(F{=}4 \to 5') + 91\,{\rm MHz} - 91\,{\rm MHz} \\ &\simeq f(F{=}4 \to 5') \qquad (3.3) \\ f_{\rm clearing} &= f_{\rm SEO} - f_{\rm clearingAOM} \end{aligned}$$

[2]The recoil velocity $v_{\rm r}$ for photons with wavelength $\lambda = 852.356$ nm is $h/(m_{\rm Cs}\lambda) = 3.5224$ mm/s.
[3]The recoil temperature for cesium is $m_{\rm Cs}v_{\rm r}^2/k_{\rm B} = 200$ nK.

$$\begin{aligned} &\simeq f(F=4 \rightarrow 5') + 91\,\text{MHz} - 91\,\text{MHz} \\ &\simeq f(F=4 \rightarrow 5') \end{aligned} \tag{3.4}$$

The clearing beam acts like a second probe beam but instead of scattering photons to be detected, the clearing beam is used to reduce the background signal by pushing all of the atoms in the $F=4$ ground state out of the detection region.

3.1.6 Magnetic sublevel-sensitive detection

The third beam overlapped with the probe and clearing beams is the Doppler-free (DF) Raman beam. It is used to transfer atoms between the $F=3, m_F=0$ and $F=4, m_F=0$ states. It comes from an 850 nm laser diode stabilized with a grating in the Littrow configuration [42]. The layout for this laser is shown in Figure 3.8. After some of the output power is split off to an optical spectrum analyzer and an rf photodiode, the light passes through an AOM (the DF Raman AOM) and a pinhole spatial filter before being overlapped with the clearing beam. Because this diode beam is overlapped with the clearing beam, it will have roughly the same size, a Gaussian beam diameter of $\sim$24 mm at the atoms. Using the beatnote between this laser and the *SEO* laser to feed back to the position of the grating, the DF Raman output frequency is locked $\sim$3.6 GHz above the *SEO* laser frequency.

$$\begin{aligned} f_{\text{DFRaman}} &= f_{\text{SEO}} + f_{\text{DFRamanVCO}} + f_{\text{DFRamanXTAL}} + f_{\text{DFRamanAOM}} \\ &\simeq f_{\text{SEO}} + 3.6\,\text{GHz} + 16\,\text{MHz} + 80\,\text{MHz} \\ &\simeq f(F=4 \rightarrow 5') + 3.7\,\text{GHz} \end{aligned} \tag{3.5}$$

By combining the dc current to this laser with an 1.3 mW microwave signal, the current to this laser is modulated at $f_{\text{mod}} = 4.6$ GHz, approximately half of cesium's ground state hyperfine splitting of 9.192 631 770 GHz. If the magnitude of the electric field is given by $E(t) = \frac{1}{2}E_0 \exp(i\omega_c t)+\text{c.c.}$, this modulation at frequency ω_{m} produces sidebands in frequency space according to the expression

$$E(t) \to \frac{E_0}{2} \sum_{n=-\infty}^{+\infty} J_n(M) \exp[i(\omega_{\mathrm{c}} + n\omega_{\mathrm{m}})t] + \mathrm{c.c.} \tag{3.6}$$

where M is the modulation depth. By varying the amplitude of the microwave signal, M is set so that the strength of the carrier and first-order sidebands are approximately equal. With the absolute laser frequency set to f_{DFRaman}, the first-order sidebands have frequencies

$$\begin{aligned} f_{+1} &= f_{\mathrm{DFRaman}} + f_{\mathrm{mod}} \\ &\simeq f(F\!=\!4 \to 5') + 91\,\mathrm{MHz} + 3.6\,\mathrm{GHz} + 4.6\,\mathrm{GHz} \qquad (3.7) \\ f_{-1} &= f_{\mathrm{DFRaman}} - f_{\mathrm{mod}} \\ &\simeq f(F\!=\!4 \to 5') + 91\,\mathrm{MHz} + 3.6\,\mathrm{GHz} - 4.6\,\mathrm{GHz} \qquad (3.8) \end{aligned}$$

separated by $f_{+1} - f_{-1} = 4.6 - (-4.6) = 9.2$ GHz, the ground state hyperfine splitting. Thus, because of the modulation, the two first-order sidebands are nearly two-photon resonant. By tuning the modulation frequency with respect to a stable frequency reference, we can make this laser resonantly drive two-photon off-resonant Raman transitions. Because both effective Raman frequencies are copropagating, the resonance is first-order insensitve to Doppler and recoil shifts. Thus, we need only cancel any ac-stark shifts due to the DF Raman laser or any remnant magnetic field shifts. With a final output power of ~ 2.2 mW, we typically achieve effective two-photon Rabi frequencies of ~ 200 Hz and detunings of around -80 Hz from Doppler-free resonance. Because these resonant sidebands are detuned ~ 1 GHz from the single-photon $F\!=\!4 \to F\!=\!5'$ transition, the resonance width is not limited by the excited state lifetime but only by the spectral width of the ~ 2.5 ms long π-pulses. This linewidth of roughly $1/(2\pi T_\pi) = 64$ Hz is much smaller than the Zeeman shift of ~ 50 kHz for magnetic bias field strengths of 72 mG. Thus, by tuning the two-photon frequency, each of the Zeeman sublevels can addressed individually.

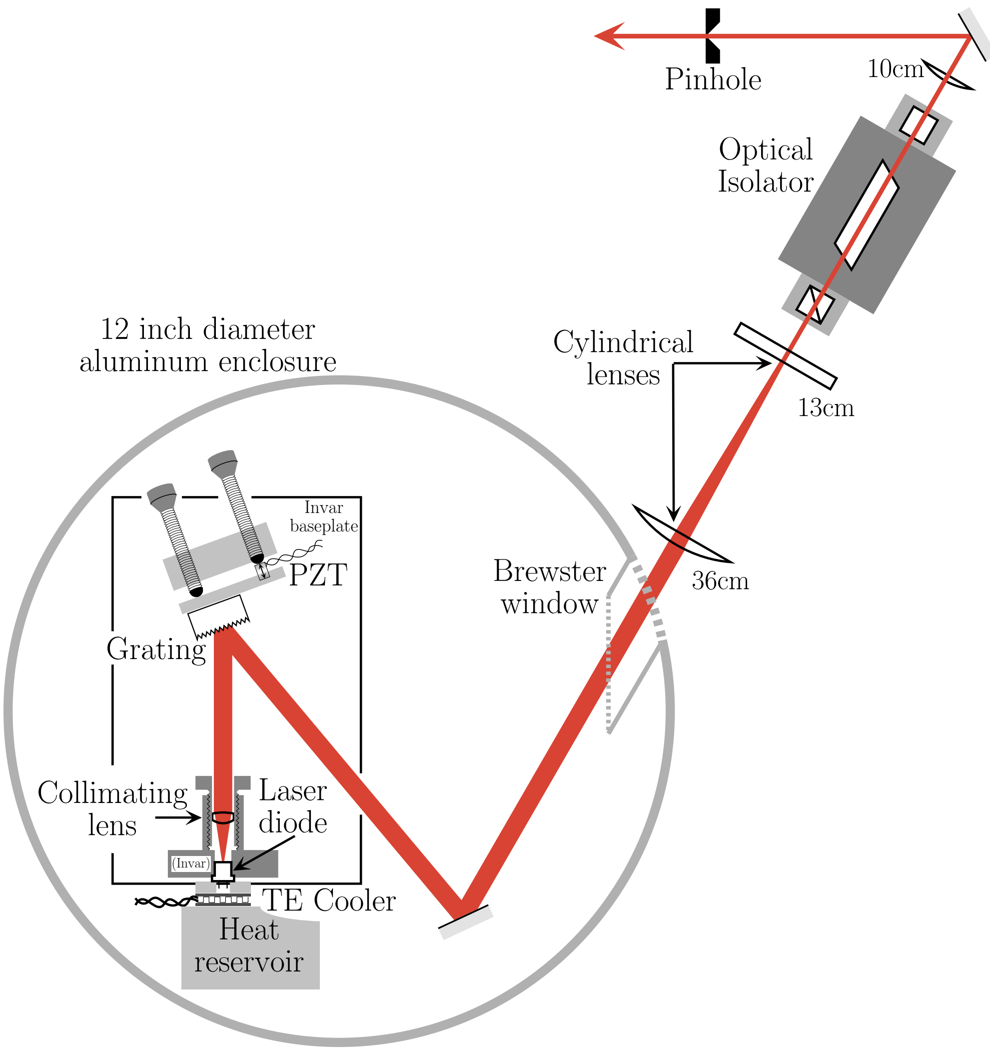


Figure 3.8: **Setup for an external cavity laser diode** used for the repumping, DF Raman, and tracer lasers. An anti-reflection (AR) coated semiconductor laser diode is pressed into an Invar block, onto which is mounted a collimating lens whose distance to the diode can be sensitively adjusted. Feedback electronics stabilize the diode's temperature by reading the signal from a thermistor (not shown) and controlling the current to a thermo-electric (TE) cooler. A grating reflects the laser output into several diffraction orders. Using a stable aluminum mirror mount, the grating is oriented so that a first order diffraction reflects exactly back into the laser. According to the Bragg condition, the reflection angle of this first order diffraction depends on the laser wavelength, so this back reflected light stabilizes the laser's output frequency. By controlling the voltage to a piezo-electric transducer (PZT) which sensitively adjusts the distance from the diode to the grating, the laser frequency can be fine tuned or locked to an atomic reference. The primary zero-order reflection continues on out of the airtight aluminum enclosure through two cylindrical lens which remove the beam's asymmetry. An optical isolator prevents downstream optics from scattering light back into the diode and disturbing its stability. Finally, the beam is focused through a pinhole with a tranmission efficiency of roughly 50% to filter out higher order spatial modes.

With the probe alone, we count *all* of the atoms in the $F\!=\!4$ state. However, by introducing spectrally narrow transitions between specific Zeeman sub-levels, we are able to detect only those atoms in the $m_F\!=\!0$ Zeeman sublevel. To detect the number of atoms launched from the trap in the $F\!=\!4, m_F\!=\!0$, for example, we would perform one Doppler-free π-pulse to transfer the atoms from the $F\!=\!4, m_F\!=\!0$ to the $F\!=\!3, m_F\!=\!0$ state. Because of the presence of the $F\!=\!3 \rightarrow F\!=\!4'$ repumping light overlapped with the trap beams, all atoms leave the trap in the $F\!=\!4$ state. Thus, after one DF Raman π-pulse, the only atoms in the $F\!=\!3$ state are atoms that were launched in the $F\!=\!4, m_F\!=\!0$ state. We then apply a clearing pulse to remove any atoms leftover in the $F\!=\!4$ state. A second Doppler-free π-pulse from the DF Raman laser brings the $F\!=\!3, m_F\!=\!0$ atoms back to the $F\!=\!4, m_F\!=\!0$ state, where they can be detected by the ordinary $F\!=\!4 \rightarrow F\!=\!5'$ probe light. The light scattered from this probe pulse will be proportional to the atoms leaving the trap in the $F\!=\!4, m_F\!=\!0$ state.

3.1.7 Zeeman pumping

To increase the final signal we magnetically polarize the atoms after they leave the trap but before they enter the magnetic shielding. To accomplish this, we turn on a laser beam tuned to the $F\!=\!4 \rightarrow F\!=\!4'$ transition. This laser enters the chamber $\sim\!5$ cm above the trap, travels horizontally through the chamber, hits a mirror, and retroreflects exactly back on itself. It is linearly polarized in the vertical direction. Because its polarization is roughly parallel to the magnetic bias field it will drive all transitions $F\!=\!4, m_f \rightarrow F\!=\!4', m_f'$ except when $m_f = 0$. After roughly 30 ns the atoms will spontaneous emit a photon and fall back into one of the states $F\!=\!4, m_f\!-\!1$, $F\!=\!4, m_f$, or $F\!=\!4, m_f\!+\!1$ with probabilities given by the angular matrix elements for that transition. Once an atom reaches the $F\!=\!4, m_F\!=\!0$ ground state, it will no longer be coupled to the excited state by the laser, and it will thus remain in that state. Because of the chance that an atom in the $F\!=\!4'$ excited state might fall into the $F\!=\!3$ ground state, out of resonance with the Zeeman pumping laser, we overlap with this beam some of the $F\!=\!3 \rightarrow F\!=\!4'$ light from the repumping laser. When the atoms encounter these beams on their way upward, they are pumped to the $F=4, m_F=0$ state. Although the $m_F=0$ signal from a sample of atoms equally distributed among the 9 possible Zeeman sublevels should increase by a factor of 9 when it encounters this "Zeeman pumping" beam, we typically see an enhancement of only 3. This is most likely due to heating as the atoms spontaneously re-emit photons. The heated atom cloud expands faster and thus fewer atoms remain in the probe beam when the detection pulse occurs. From Figure 3.7, the spread of the atomic velocities in the vertical direction increases from 1.40 cm/s to 1.64 cm/s. Subtracting these two rms velocities in quadrature indicates that the Zeeman pumping adds a velocity of 0.85 cm/s, equivalent to 2.4 single photon recoils or a temperature increase of 1.2 μK.

The Zeeman pumping beam comes originally from the *SEO* Ti-Sapphire laser but passes through two AOMs before entering the chamber (see Figure 3.5). The first Zeeman pumping AOM at frequency $f_{\mathrm{ZP1}} = 117.5$ MHz is on all the time while the second Zeeman pumping AOM at frequency $f_{\mathrm{ZP2}} = 108$ MHz switches on 1 ms after the launch, well before the atoms pass through the Zeeman pumping beam.

$$\begin{aligned} f_{\mathrm{ZP}} &= f_{\mathrm{SEO}} - 2f_{\mathrm{ZP1}} - f_{\mathrm{ZP1}} \\ &\simeq f_{\mathrm{SEO}} + 91\,\mathrm{MHz} - 235\,\mathrm{MHz} - 108\,\mathrm{MHz} \\ &\simeq f(F=4 \rightarrow 5') - 252\,\mathrm{MHz} \\ &\simeq f(F=4 \rightarrow 4') \end{aligned} \tag{3.9}$$

Also at this time, the MOT trim coils switch current levels to a setting which applies a large bias field in the vertical direction. Because the final measurement will involve magnetic sublevel sensitive detection, we must preserve the magnetic dipole orientation of the atoms by applying a well-defined magnetic bias field. For the Zeeman pumping process to work properly, this magnetic field must be vertical to match the polarization of the Zeeman pumping beam. It must also be in the same direction as the magnetic bias field within the magnetic shielding so that there is no point along the atoms' trajectory where the magnitude of the field vanishes.

3.2 Adiabatic passage beam generation

The most fundamental part of the entire experiment consists of the Raman beams which are used to construct the atom interferometers and thereby measure the recoil shift. These beams must have precise and well defined wavefront properties. They must have stable absolute frequency with respect to the cesium atom. They must be split into two counter-propagating beams whose phase and frequency difference are ultra-stable with respect to a precision time standard even when changed to compensate for Doppler and recoil shifts. In order to adiabatically transfer atoms between internal states, we must be able to independently control the intensity of each beam. Finally, we must be able to electronically switch the beam direction.

3.2.1 Laser source

As diagrammed in Figure 3.9, the adiabatic transfer or "Raman" beams originate from a Model 599 titanium-sapphire ring laser from *Coherent* pumped by 11 W from the same large frame Argon Ion laser that pumps the *SEO* Ti-sapphire. Although the ring cavity is in the vertical plane instead of the horizontal plane, this laser contains the same components described in the layout for the *SEO* laser (Figure 3.3). As shown in Figure 3.9, $\sim 20mW$ of the 900 mW output at 894.6 nm is split off, 10 mW to an external lock to cesium and 10 mW to a wavemeter which determines the wavelength to ± 0.0005 nm. The remainder of the power goes to the generation of the two optical frequencies used to adiabatically transfer atoms between the $F=3$ and $F=4$ ground states and thereby build the atom interferometers. From here on, these two frequencies addressing the $F=3 \rightarrow F=3'$ and $F=4 \rightarrow F=3'$ transitions of the D1 line of cesium at 894.606 nm will be called "$F=3$" and "$F=4$", respectively.

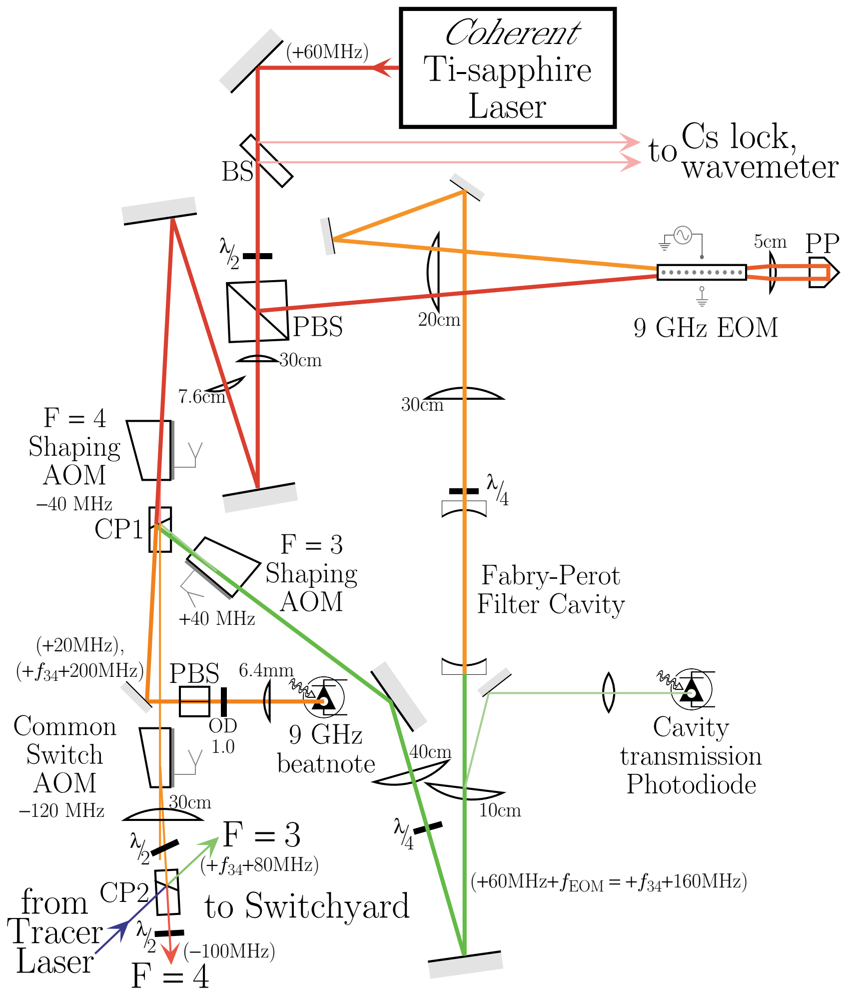


Figure 3.9: **Preparation of the adiabatic passage beams.** The laser beams used to drive two-photon transitions and build the interferometers come from a *Coherent* Titanium-sapphire laser. A small fraction of its output is immediately separated with a power beam splitter (BS) in order to measure the laser's wavelength and to lock the laser to cesium. Roughly 80% of the remaining power is split off by a polarizing beamsplitter (PBS) and frequency shifted by 9 GHz. This microwave frequency shift is achieved by first phase modulating the light with a 9 GHz electro-optic modulator (EOM) and then using a Fabry-Perot cavity to filter out all spectral components except the desired sideband. In order to achieve the required modulation depth from the EOM, a Porro prism (PP) reflects the light back through the EOM. This "$F=3$" light is overlapped with the other 20% of the laser power, the "$F=4$" light, at the first calcite polarizer (CP1). A microwave photodiode detects the 9 GHz beat frequency between these two lasers, allowing their phase and frequency difference to be precisely controlled. To control the intensity of each laser, an acousto-optic modulator (AOM) is placed in each path before CP1. In addition to these "shaping" AOMs, a "common switch AOM" is placed between CP1 and CP2. By varying the amplitude of the rf signal to these three AOMs, the fraction of light diffracted toward the switchyard and ultimately to the atoms can be controlled. The laser frequencies relative to the $F=4$ to $F=3'$ D1 transition are given in parentheses ().

3.2.2 Second optical frequency

To generate the two optical frequencies, we lock the Ti-sapphire laser near the $F=4$ transition and shift a fraction of the output by 9.2 GHz to address the $F=3$ transition. As shown in Figure 3.9, we use an electro-optic modulator (EOM) to produce the microwave frequency shift of f_{EOM}. This EOM is a prototype design[4] from *New Focus* which consists of 0.6 mm wide by 0.4 mm tall by 33.3 mm long anti-reflection (AR) coated $LiTaO_4$ crystal housed in a specially designed case. The case forms a cavity that resonates at certain rf and microwave frequencies, one of which is 9 GHz. The geometry of this enclosure is designed so that when microwave power is properly coupled into the cavity, the electric field produced in the crystal will be resonantly enhanced. According to the Pockels effect, this electric field alters the real part of the index of refraction [43]. Consequently, light whose polarization is aligned with the electric field in the crystal will experience a phase shift. Because this phase shift is proportional to the electric field, the phase of the light will be effectively modulated at the microwave frequency. As mentioned in equation (3.6), in frequency space this modulation produces sidebands separated by the modulation frequency. Even with the long crystal and the resonantly enhanced electric field driven with input microwave power of 1.0 W, we are not able to obtain the optimum modulation depth with a single pass. By using a square-cut knife-edge Porro prism to retro-reflect the light back through the crystal, however, we are able to maximize the amount of power in the first-order sidebands, so that they each account for roughly one third of the optical power, with the remaining third of the power divided between the carrier and higher order sidebands. We are interested in only one of the sidebands. In order to filter out the other unwanted frequency components, we next pass the modulated light through a Fabry-Perot cavity with a finesse of ~ 150 and an off-resonant transmission efficiency of ~ 0.015. By tuning this Fabry-Perot cavity to resonate at only one of the frequencies, we select a single sideband. We obtain an optical transmission of more than 97% for the first pass through the EOM crystal and a total transmission of 89%

[4] A modified version of this prototype design which we helped construct is now sold by *New Focus* as part of their 484X-series.

for both passes[5]. The resonant transmission efficiency through the Fabry-Perot cavity is 80%. Thus, the process of shifting an optical frequency by a microwave frequency f_{EOM} has a total efficiency of $\sim$24%.

In order to obtain approximately the same amount of power in the $F=3$ and $F=4$ beams, an adjustable power splitter separates the Ti-sapphire light into fractions of roughly 80% and 20%, with the larger amount going to the frequency shifting section which produces the $F=3$ beam. This power splitter is simply a zero-order $\lambda/2$-plate plus a polarizing beamsplitter (PBS) cube. After shifting the frequency of the $F=4$ light to produce the $F=3$ beam, we now have effectively two 170 mW lasers whose frequency difference is given by the frequency of the microwave signal driving the EOM. We are now ready to control the intensity and direction of these two lasers.

3.2.3 Shaping AOMs

To control the intensity, we pass each of these beams through its own 40 MHz AOM made by *Andersen Laboratories*. By varying the amplitude of the $f_{shAOM} = 40$ MHz rf signal to these two "shaping AOMs" we can control the fraction of light they diffract into the first-order. These anti-reflection coated shaping AOMs transmit $> 99\%$ of the light. At an optimum rf power of $\sim$3 W, they have diffraction efficiencies of 90% and 97% for the $F=3$ and $F=4$ beams, respectively, By controlling the amplitude of the two separate rf signals, we can thus electronically vary the intensity of the light sent to the atoms.

The outputs of the two shaping AOMs are overlapped at the first calcite polarizer (CP1 in Figure 3.9). To improve the overlap efficiency, we use a zero-order $\lambda/4$-plate to set the polarization of the $F=3$ beam to be linear and perpendicular to the plane of the optical table (S-polarized). A $\lambda/4$-plate instead of a $\lambda/2$-plate is required because the polarization of the light is converted from linear to circular by another $\lambda/4$-plate placed just before the Fabry-Perot optical filter cavity. This first $\lambda/4$-plate serves to minimize the light that reflects off of the mode-matched Fabry-Perot cavity and travels back into the Ti-sapphire laser thus interfering with the laser's frequency

[5]We estimate there is some loss due to slight clipping on the horizontal edges of the crystal.

stability. At the overlapping polarizer CP1, the polarizations of the $F=3$ and $F=4$ are therefore nearly orthogonal.

3.2.4 Common switch AOM

The first-order light from the shaping AOMs continues on through an *Isomet* model 1206C-1-830 AOM. This “common switch AOM” is driven by a 2.0 W rf signal at $f_{swAOM} = 120$ MHz. The amplitude of this 120 MHz signal is controlled by a (*Mini-Circuits* model ZYSW-2-50DR) rf switch which leaves the rf on or turns it off with an isolation of 58 dB. Because at this point both the $F=3$ and $F=4$ beams are overlapped and spatially mode matched, the common switch AOM serves to switch both light fields on or off together in exactly the same manner with exactly the same phase shift. This AOM was added to the setup partway into the experiment to fix a previously unexplained systematic error from the $\pi/2$-pulses (see Section 6.7), so much of the recoil data were taken without it.

3.2.5 Switchyard

After the common switch AOM, the overlapped but orthogonally polarized $F=3$ and $F=4$ beams are separated by a second calcite polarizer (CP2 in Figures 3.9 and 3.10). When aligned to the first polarizer, CP2 separates the two orthogonal Raman polarizations with leakage less than 1×10^{-5}. With the common switch AOM in the beam, however, this isolation decreases to roughly 10^{-3}, most likely due to small (possibly thermally induced) birefringent properties of the AOM crystal. To repurify the polarization we insert an additional zero-order $\lambda/2$-plate designed for use at 852 nm just before CP2. By adjusting the angle of incidence and angle of the optic axis of this waveplate, we are able to compensate for the effect of the common switch AOM and return the leakage to $\sim10^{-5}$. However, as might be expected from thermally induced birefringence, this cancelation does not remain perfect, causing the leakage to drift up to but never higher than $\sim1\times10^{-4}$. This second calcite polarizer CP2 also serves as the point where the tracer beam is overlapped with the Raman beams. The tracer beam will be discussed in Section 3.3.3.

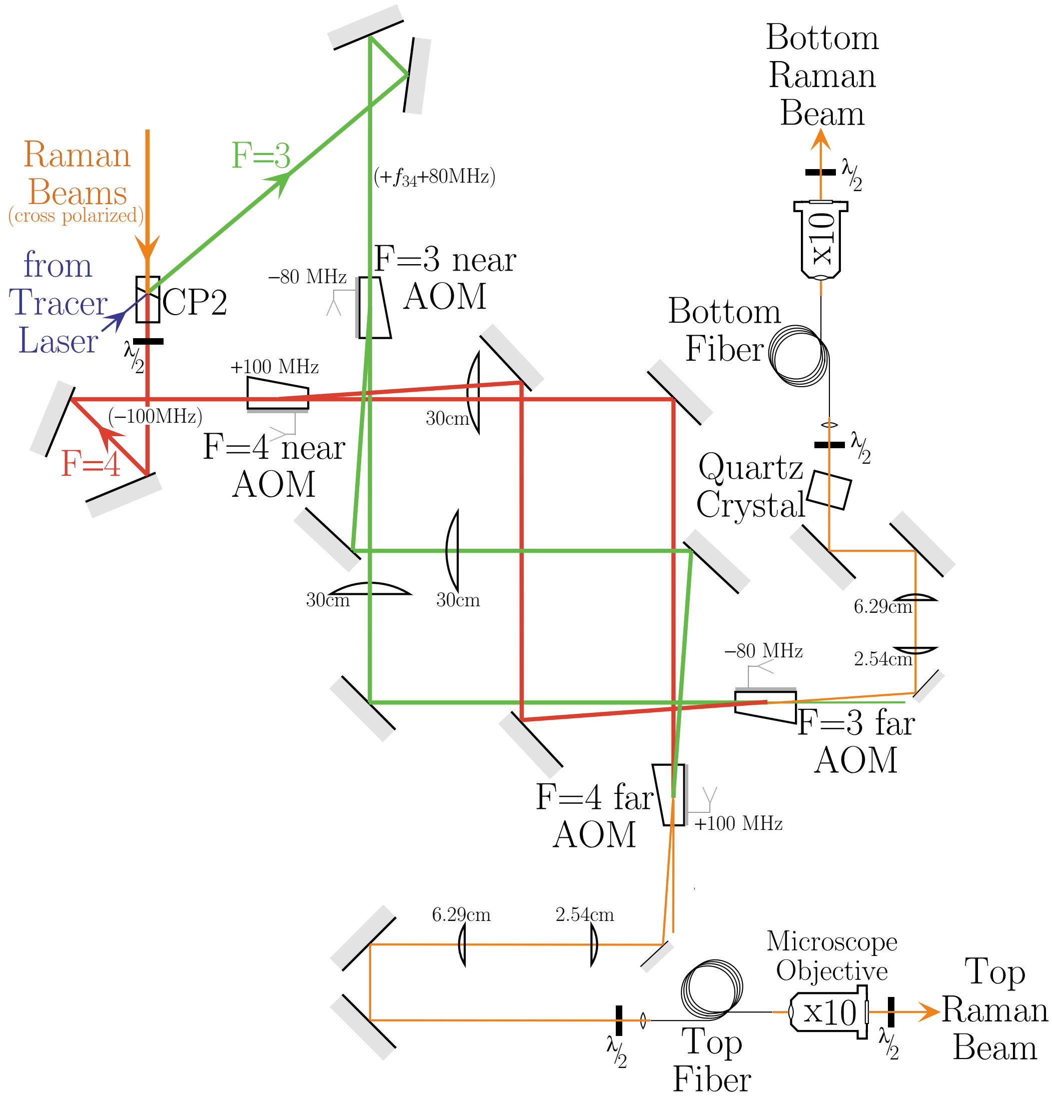


Figure 3.10: **The Raman beam switchyard** allows us to electronically reverse the direction of the Raman lasers. Two pairs ("near" and "far") of acousto-optic modulators (AOMs) selectively direct each of the two input frequencies ("$F=3$" and "$F=4$") into either of the two possible output directions ultimately pointed toward the top and bottom of the vacuum chamber. If the switchyard is on, each laser is diffracted by only one switchyard AOM into one of the two optical fibers. The fibers are used to 1) improve the isolation of all of the AOMs and 2) to provide the final spatial filtering before the beams interact with the atoms. The "tracer" laser is overlapped with the Raman lasers at the second calcite polarizer (CP2) and thus propagates orthogonally polarized relative to the Raman beams. A quartz crystal is inserted before the bottom fiber to rotate the polarization of just the tracer beam so that its polarization is then approximately parallel to that of the Raman beams. The laser frequencies at various points relative to the $F=4$ to $F=3'$ D1 transition are given in parentheses ()s. Note that the Raman light emerging from the fibers is resonant with either the $F=3 \to 3'$ or the $F=4 \to 4'$ transition.

After the beams split they enter the "switchyard" which consists of four *Isomet* model 1205C-1-830 AOMS, a few refocusing lenses, and several mirrors (see Figure 3.10). The switchyard has two input paths, labeled $F=3$ and $F=4$, and two output paths, labeled "top" and "bottom". Its purpose is to allow us to electronically switch between three possible conditions: 1) "normal" condition where $F=3$ goes to bottom and $F=4$ goes to top, 2) "invert" condition where $F=3$ goes to top and $F=4$ goes to bottom, and 3) "off" condition when no light emerges into either output. The switchyard thus allows us to change the beam direction and also provides an additional layer of isolation to ensure that the Raman light is really off when it is supposed to be.

To better understand how the switchyard works, consider the normal condition. In this case, the "near" AOMs (see Figure 3.10) are off and the "far" AOMs are on. The $F=4$ light, for example, passes unshifted through the $F=4$ near AOM but is then shifted into the top fiber by the $F=4$ far AOM, and similarly for the $F=3$ beam into the bottom fiber. In the inverted condition the control electronics switch the rf signals so that the near AOMs are on and the far AOMs are off. In this case, the $F=4$ light is shifted by the first AOM it encounters (the $F=4$ near AOM) into a path that when the next two mirrors are correctly oriented overlaps with the path that the $F=3$ far AOM would shift the beam into if it were on (i.e. the path headed into the bottom fiber). In a similar way, the $F=3$ near AOM directs the $F=3$ beam through the $F=4$ far AOM, which must be off, and into the top fiber. Note that the condition with all four switchyard AOMs on is not permitted. The OFF condition is when all switchyard AOMs are off.

3.2.6 Spatial filtering

After emerging from the switchyard, each interferometer Raman beam passes through a *3M* FS-PM-4611HT single-mode polarization-preserving optical fiber. A high numerical aperture lens focuses the light into the 5.3 μm mode field diameter of the fiber. A zero-order $\lambda/2$-plate before the lens sets the polarization of the beam to match the orientation of the asymmetry of the fiber core. Because the fiber input and output facets are not anti-reflection coated or angle polished, they tend to exactly retroreflect a fraction of the light. The fiber thus acts like an extremely thick glass etalon, whose transmission efficiency varies as the reflections off its input and output faces interfere. To minimize this effect we insert some index-matching gel made by *Math Associates* between the input focusing lens and the bare fiber. With the gel in place, the light sees roughly the same index of refraction as it passes through the focusing lens, propagates through the gel, and into the glass fiber. Because the gel changes the effective focal length of the focusing lens, we compensate for this by re-optimizing the distance between the lens and the fiber.

After emerging from the ~ 1 m long bare fibers the beams diverge rapidly in free space until they are focused by $\times 10$ microscope objectives to a Gaussian beam waist diameter of 114 μm. Including the input focusing lens and this microscope objective after the output, we measure a total transmission efficiency of 60% through the top fiber and 40% through the bottom fiber[6]. This efficiency could be improved by as much as 15% by AR coating the fibers and replacing the microscope objectives, which are made for use at visible frequencies and are not optimized for transmission at 894.6 nm. The total transmission efficiency will ultimately be limited by the input beam quality. Although the spatial mode emerging from the Ti-sapphire laser is quite good, after being diffracted by three AOMs (one shaping, the common switching, and one switchyard), the final beam may not be as easily matched with the mode of the fiber. In fact, mode quality is the primary reason for using optical fibers. The recoil measurement depends on the atoms interacting with lasers of extremely well defined momentum, which is defined by the local wavefront gradient. By filtering out higher order spatial modes, the fibers insure that the interferometer beams have clean and well-defined wavefronts. In addition to spatial filtering, the fibers also improve the on-off insolation of the AOM intensity switches (see Table 3.1).

[6]The difference in transmission efficiency is probably due to the quality of the fiber facets which were cleaved using a precision fiber cleaver from *Fujikura*.

Table 3.1: **Isolation performance of the switchyard.** In order to pass through one of the two optical fibers, the $F = 4$ and $F = 3$ Raman beams must be diffracted in series by three acousto-optic modulators (AOMs): the individual shaping AOMs (Ind), the common switch AOM (Com), and one AOM from the switchyard. The state of these AOMs is represented as either on (1) or off (0). The state of the switchyard is determined by two switchyard controls (10=normal, 01=inverted, and 00=off). The infinity ∞ symbol indicates that the light emerging from the fiber saturated the sensitive photodiode we used to detect the leakage signals. All other numbers represent the amount of optical power emerging from the fiber relative the fully on level.

		Switchyard		**Top Fiber**		**Bottom Fiber**	
Ind	Com	Norm	Inv	$F = 4$	$F = 3$	$F = 4$	$F = 3$
1	1	1	0	∞	3×10^{-5}	2×10^{-5}	∞
0	1	1	0	4×10^{-4}	1×10^{-7}	3×10^{-7}	2×10^{-4}
1	0	1	0	8×10^{-6}	9×10^{-9}	6×10^{-9}	8×10^{-6}
0	0	1	0	3×10^{-9}	$< 6 \times 10^{-10}$	$< 6 \times 10^{-9}$	2×10^{-9}
1	1	0	1	2×10^{-5}	∞	∞	3×10^{-5}
0	1	0	1	2×10^{-7}	1×10^{-4}	4×10^{-4}	1×10^{-7}
1	0	0	1	9×10^{-9}	6×10^{-6}	1×10^{-5}	8×10^{-9}
0	0	0	1	$< 6 \times 10^{-10}$	1×10^{-9}	4×10^{-9}	$< 6 \times 10^{-10}$
1	1	0	0	6×10^{-8}	6×10^{-7}	1×10^{-7}	7×10^{-7}
0	1	0	0	$< 1 \times 10^{-9}$	$< 1 \times 10^{-9}$	$< 2 \times 10^{-9}$	$< 2 \times 10^{-9}$
1	0	0	0	3×10^{-9}	3×10^{-9}	$< 6 \times 10^{-10}$	1×10^{-9}

3.2.7 Collimation and polarization

After emerging from the fibers, the light is collimated, circularly polarized, and then directed vertically into the vacuum chamber. After being focused by the microscope objectives at the outputs of the fibers each beam is allowed to expand freely to a Gaussian beam diameter of 1.91 ± 0.15 cm before being collimated by a plano-convex lens of focal length 2 m (part number PLCX-50.8-1030.2-C from *CVI*). Each beam then reflects off three more high quality dielectric mirrors, which direct it into the vacuum chamber. Except for the last top mirror, all of the mirrors are at least 3 inches in diameter. Because of space constraints, the last top mirror is cut at 45° from two inch round stock, so it is elliptical with 2 inches for its smaller dimension. Each mirror

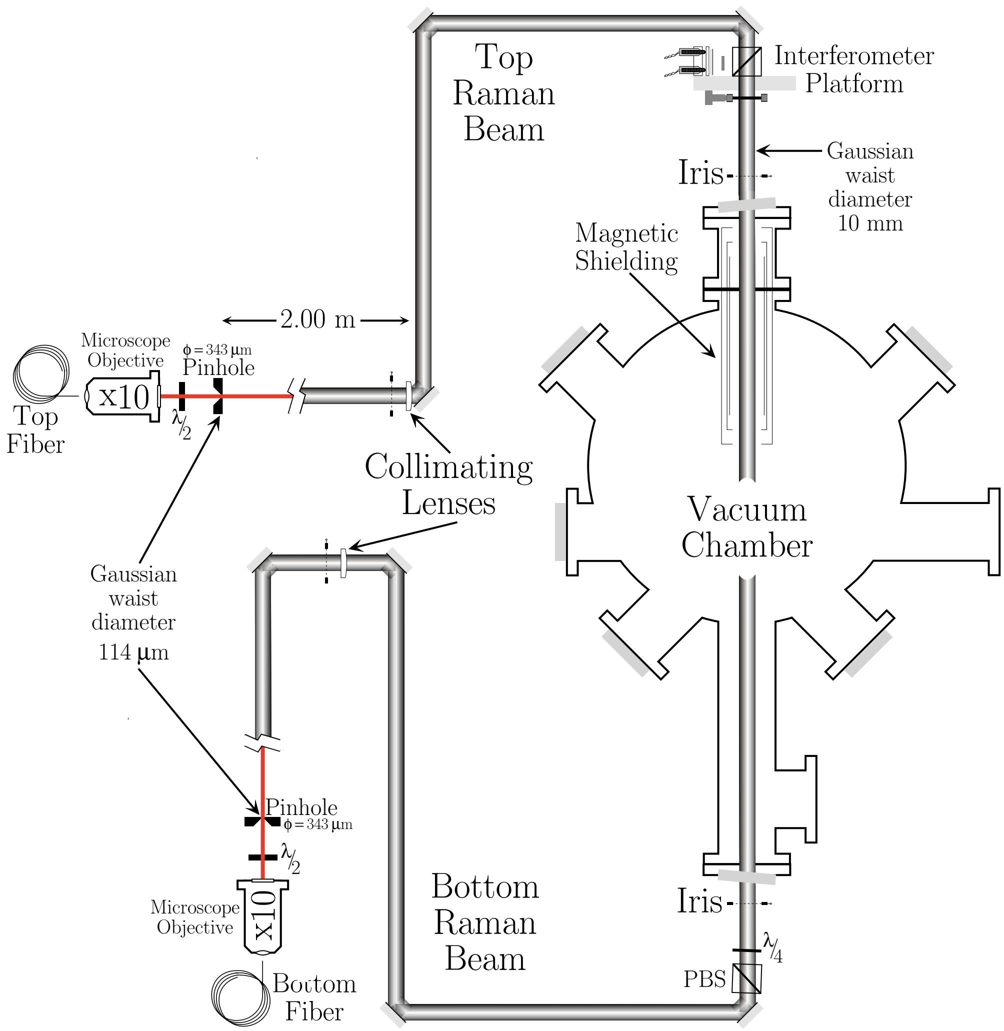


Figure 3.11: **Final Raman beam preparation**. After being spatially filtered by an optical fiber, each Raman beam is focused by a $\times 10$ microscope objective to a Gaussian waist diameter of $114\,\mu$m, where its position is determined using a $343\,\mu$m diameter pinhole. Each beam expands to a diameter of 9.7 mm before it is collimated by a $f = 2$ m plano-convex lens. High quality optics then direct the beams to the vacuum chamber. Before entering the chamber, each beam passes through a polarizing beamsplitter (PBS) cube that purifies its polarization and a $\lambda/4$-plate that converts its polarization to circular. Irises above and below the chamber fix the position of the beam. Overlapped with each Raman beam is the tracer laser. The beat signal between the bottom and top tracer beams is detected by the optical interferometer on the interferometer platform (see Figure 3.14) and used by the tracer PLL to remove phase noise from the Raman beams. Not shown is the active vibration isolation system (see Figure 3.15) that stabilizes and isolates the interferometer platform from environmental vibrations. By amplitude modulating the tracer laser and comparing the phase delay of this modulation signal for the top path to the same signal from the bottom path, we estimate that the optical path from CP2 (see Figure 3.10) to the top PBS cube is no more than 12 cm longer for the top path than for the bottom path.

is specified to be $\lambda/10$ at 633 nm with scratch-dig rating of 10-5. Before entering the chamber, each beam passes through a 2 inch polarizing beam splitter cube and then a zero-order $\lambda/4$-plate with 2 inch clear aperture from *Special Optics*. The polarizing cubes clean up the polarizations of the already roughly linearly polarized beams before the $\lambda/4$-plates converts their polarization from linear to circular. Although we do not know the actual sign of the helicity used in the lab, without loss of generality, from here on we will assume the Raman beams are $\hat{\sigma}_+$ polarized. To minimize the chance of the bottom beam reflecting off the surfaces of the $\lambda/4$-plate or the polarizing beam splitter cube, each of these optics is tilted from normal so that the small amount of reflected light does not make it back to the atoms inside the vacuum chamber. For the same reason, as depicted in Figure 3.11, the top and bottom windows of the vacuum chamber are also tilted at a 5° from normal.

3.3 Frequency and phase control

3.3.1 Difference frequency

After the first calcite polarizer CP1 overlaps the $F\!=\!3$ and $F\!=\!4$ polarizer, a mirror picks off just the zero-order light from the shaping AOMs and directs it toward a microwave photodiode. Because the orthogonally polarized light beams will not interfere and produce a beatnote, a polarizing beamsplitter cube oriented at 45° projects roughly half of each frequency component into the same linear polarization $\sim 45°$ from S-polarized. These two overlapped beams are then focused by a 6.4 mm focal length lens onto a 25 μm diameter photodiode. This gallium-arsenide photodetector is a custom-made design of *Agilent* (formerly *Hewlett Packard*) [44]. For our beams focused to a Gaussian beam waist diameter of 54 μm, this detector has a sensitivity of 0.1 A/W. Via an SMA adaptor the photodetector is connected directly to a *Picosecond Pulse Labs* bias tee (model 5550B) which allows the incoming dc bias voltage to be separated from the outgoing microwave signal. This signal terminates in a JCA812-300 microwave amplifier from *JCA Technology* which has a specified gain and noise floor of 24 and 2.4 dB, respectively.

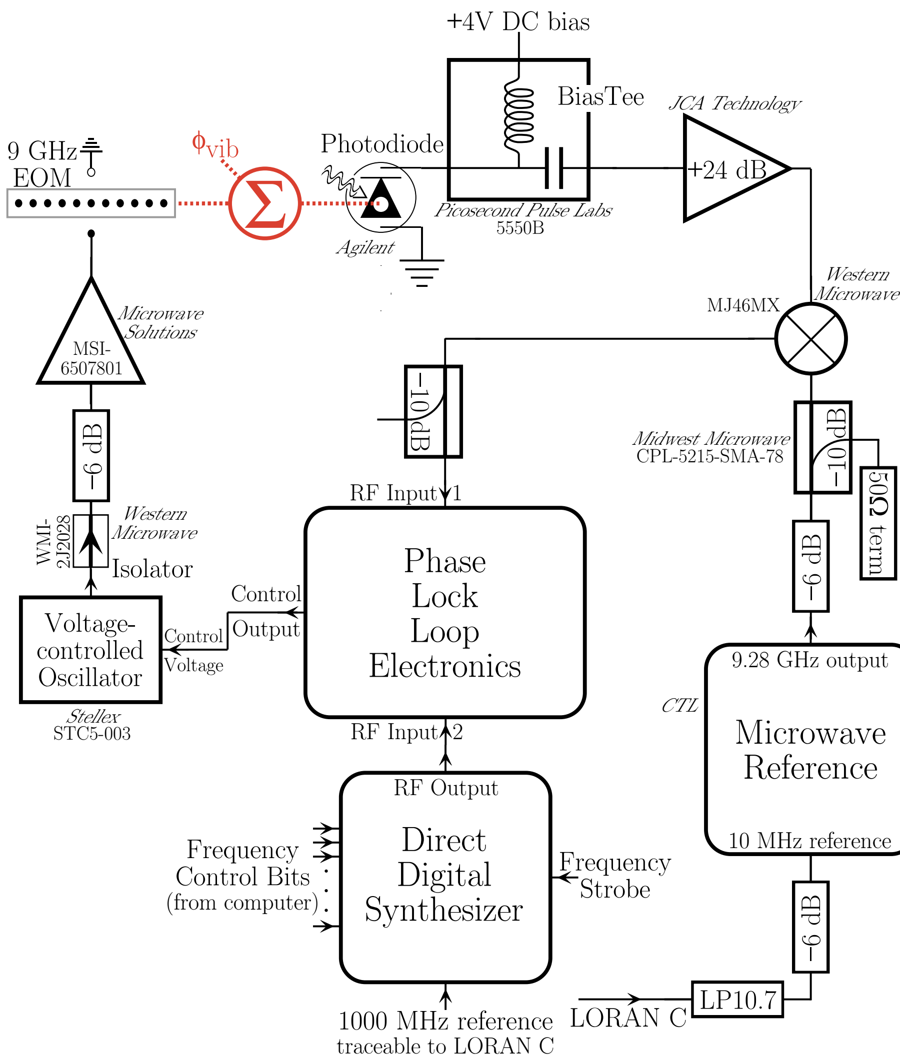


Figure 3.12: **Microwave beatnote.** A 9 GHz resonant cavity electro-optic modulator (EOM) frequency offsets the $F=3$ laser from the $F=4$ laser. A microwave photodiode detects the beat signal between these two lasers. This 9 GHz signal is mixed down to the radio-frequency (rf) regime using a precision microwave reference whose fixed frequency output is exactly 928 times its external reference. Its 10 MHz external reference is based on the LORAN C frequency standard which can ultimately be traced to the *NIST* atomic clocks. Once in the rf regime, phase lock loop (PLL) electronics discussed in Appendix B compare this beatnote with the output of a direct digital synthesizer (DDS), whose output frequency is computer controlled and also traceable to the LORAN C reference. The PLL controls the voltage-controlled oscillator (VCO) which drives the EOM, thus closing the loop. With the feedback loop closed, the frequency difference between the two Raman lasers can be precisely controlled with the DDS while their phase difference is stably locked to an atomic clock standard. The PLL thus removes all phase noise (represented by ϕ_{vib}) due to the relative motion of all optical elements up until the photodiode. The dotted lines represent the optical part of the feedback loop.

Figure 3.12 diagrams how the 9.3 GHz output of the microwave beatnote is mixed down and then used to phase lock the EOM frequency to a stable time reference. First, the output of microwave beatnote amplifier is mixed down by a *Western Microwave* MJ46MX mixer. The reference signal connected to the LO port of the mixer comes from a cw precision microwave source made by *CTI Communications*. This source contains two oscillator plus phase lock loop (PLL) stages. These two PLLs lock a dielectric resonance oscillator (DRO) at a frequency exactly 928 times a 10 MHz external reference signal. This reference signal comes directly from a *Stanford Research Systems* FS700 frequency standard, which receives the LORAN C [45] timing signal maintained by *United States Naval Observatory*. This timing signal is certified by the *National Bureau of Standards* and traceable to the atomic clock time standard maintained by the *National Institute of Standards and Technology* in Colorado. In this way our microwave reference is locked to an accurate time standard.

With the beatnote at frequency $f_{\mathrm{EOM}} \simeq f_{34} + 100\,\mathrm{MHz}$ at its RF port and the microwave reference at $9\,280\,000.000\,\mathrm{Hz} = f_{34} + 87.368\,23\,\mathrm{MHz}$ Hz at its LO port, the IF port of the microwave mixer outputs a signal whose frequency is $f_{\mathrm{EOM}} - 9.28\,\mathrm{GHz} \simeq 12.631\,77$ MHz. This rf signal is then compared with the output of the direct digital synthesizer (DDS) by the Raman PLL (see Appendix B) that controls the VCO driving the EOM. This closes the loop and thereby phaselocks the $F\!=\!3$ beam to the $F\!=\!4$ beam with a difference frequency close to cesium's ground state hyperfine splitting but precisely tunable in discrete steps of $\sim$0.233 Hz by a frequency synthesizer stable with respect to the official time standard. Because the microwave beatnote also senses any shift of phase of the $F\!=\!3$ light with respect to the phase of the $F\!=\!4$ light, the Raman PLL also removes any relative phase noise between the two lasers.

3.3.2 Absolute frequency

To determine the absolute laser frequency of the $F\!=\!3$ and $F\!=\!4$ components, we must trace the frequencies from the source through all of the frequency shifting optics until we arrive at the atoms. At the atoms the laser frequencies must be tuned to the

$F=3 \rightarrow F=3'$ and $F=4 \rightarrow F=3'$ cesium transitions, respectively.

First, we discuss Figure 3.13 which shows how the *Coherent* Ti-sapphire laser source is locked to cesium. The $\sim$10 mW split off from the main output at frequency f_{Coh} is immediately split into two beams with a controllable power ratio using a zero-order $\lambda/2$-plate followed by a polarizing beamsplitter cube. The beam which continues on through the cesium cell and into the detection photodiode will in this section be called the "probe". The other beam called the "pump" passes first through an AOM and then an EOM before being overlapped with the probe beam within the cesium cell. Because the frequency of the $f_{\mathrm{CsAOM}} \simeq 60$ MHz rf signal at the AOM is varied to control the absolute laser frequency relative to cesium, the pump beam passes twice through this AOM. Exactly retro-reflecting the pump beam after it passes once through the AOM guarantees that the return beam which is shifted twice by the AOM to frequency $f_{\mathsf{Coh}} - 2f_{\mathrm{CsAOM}}$ will not move as the diffraction angle changes with AOM driving frequency. In order to separate the exactly retro-reflected returning pump beam from the incoming pump beam without losing power, we use the beam's polarization. The incoming pump beam is S-polarized by a second polarizing beamsplitter cube. After one pass through the AOM it encounters a zero-order $\lambda/4$-plate which circularizes the beam's polarization. After this beam reflects off the retro-mirror it passes again through the $\lambda/4$-plate converting the polarization to P-polarized, which is orthogonal to the input polarization. This ongoing beam then passes through the polarizing beamsplitter cube without being deflected.

The frequency shifted pump beam next passes through an EOM crystal whose applied electric field oscillates sinusoidally at 3.53 MHz. This pump beam which is frequency shifted and now also phase modulated at 3.53 MHz by the EOM is then overlapped with the probe beam using an $R = 70, T = 30\%$ power beamsplitter. The power beamsplitter is oriented so that the pump counter-propagates with the probe beam through the cesium cell. Because the pump beam frequency $f_{\mathrm{pump}} = f_{\mathsf{Coh}} - 2f_{\mathrm{CsAOM}}$ is tuned near the $F=4 \rightarrow F=3'$ cesium D1 transition, the pump beam polarizes the cesium atoms it encounters. Because the pump beam is phase modulated, the atomic polarization is also phase modulated. When the probe

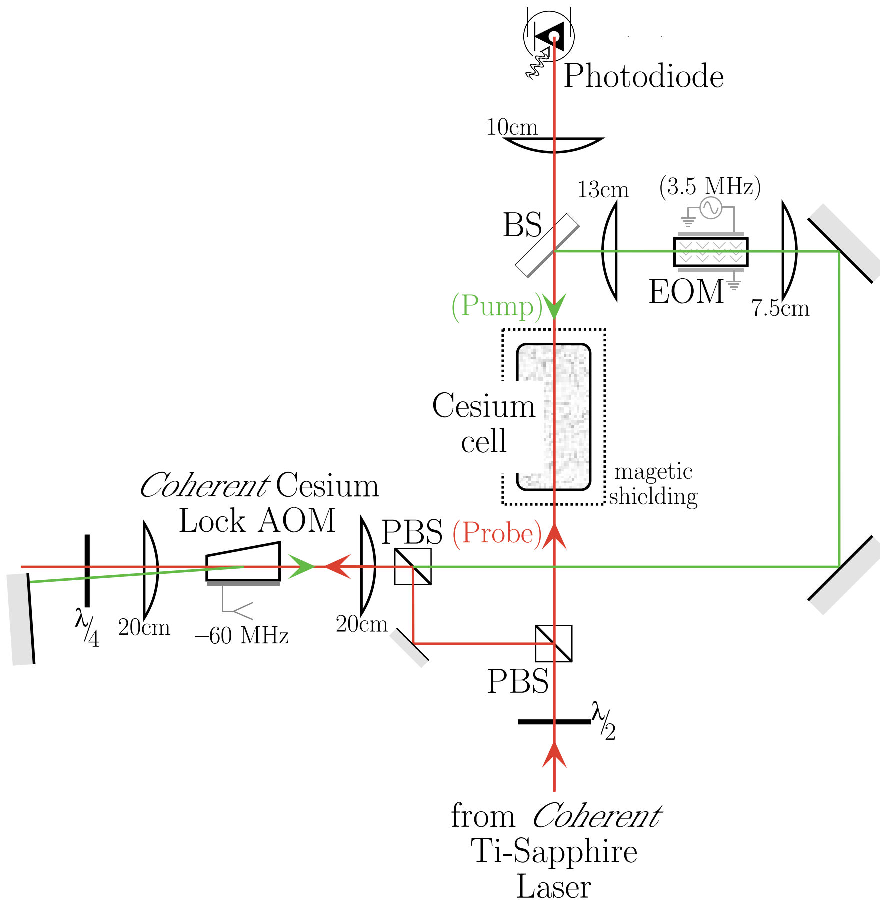


Figure 3.13: **Optical setup used to lock the *Coherent* Ti-sapphire laser to the cesium transition at 894.6 nm.** The small fraction of light separated from the laser's main output is immediately split using a polarizing beamsplitter (PBS) into a "pump" and "probe" beam. The probe beam continues on through a magnetically shielded cesium cell and then into a photodiode. The pump beam is directed twice through an acousto-optic modulator (AOM) and then through an electro-optic modulator (EOM) that modulates its phase at 3.53 MHz before it enters the cesium cell. When the laser is tuned near a cesium transition, it polarizes the atoms. Since it is created by the light, this atomic polarization also modulates with the pump beam. The probe beam interacts with these same atoms and thus also undergoes the same modulation, which can be detected by the photodiode. By mixing down the resulting electronic signal with a copy of the signal driving the EOM, a dispersive lock error signal can be generated. Because the pump and probe beams counterpropagate, the Doppler shifts of the individual cesium atoms cancel, and thus the width of the dispersive feature is close to natural linewidth of the atomic transition. The "lock AOM" shifts the frequency of the light by twice the frequency of its radio-frequency (rf) driving signal ($f_{\mathrm{CsAOM}} = -60$ MHz). Therefore, by varying f_{CsAOM}, one can control the laser's absolute frequency relative to the cesium transition. Note that because the probe beam is modulated only via the atoms, the potential for systematic lock offsets is greatly reduced. In fact, using the laser cooled atoms in the main vacuum chamber, we have verified that the long term absolute accuracy of this lock is better than ±100 kHz.

beam encounters this polarized atomic sample, it acquires the same modulation signal. Pure phase modulation would of course produce no amplitude oscillation at a photodetector. However, because the modulated light must pass through the dispersive cesium medium, the phase modulation becomes amplitude modulation whenever the laser frequency crosses one of the cesium transitions. This light amplitude modulation is converted to an electronic signal by a photodiode with sufficient bandwidth. At this point, the electronic signal is demodulated with a copy of the 3.53 MHz driving signal. As with all saturation spectroscopy signals, the lock signals are visible only when both the pump and the probe beam address the same atoms. Because the pump and probe beam counterpropagate, this occurs only when the Doppler shift for the pump and the probe beam relative to a particular atomic velocity class are equal and opposite. Thus, when locking to the $F=4 \rightarrow 3$ transition,

$$f_{\text{pump}} + f_{\text{probe}} = f_{4\rightarrow 3^0} \tag{3.10}$$

and similarly for the $F=4 \rightarrow F=4'$ transition. From Figures 3.9 and 3.13, it is clear that $f_{\text{probe}} = f_{\mathsf{Coh}}$, and since f_{pump} is shifted by the AOM, equation (3.10) becomes

$$f_{\mathsf{Coh}} - 2f_{\text{CsAOM}} + f_{\mathsf{Coh}} = f_{4\rightarrow 3^0} \tag{3.11}$$

Solving this equation for f_{Coh} gives the absolute frequency of the laser

$$\begin{aligned} f_{\mathsf{Coh}} &= f(F=4 \rightarrow F=3') + f_{\text{CsAOM}} \\ &\simeq c/(894.606\,\text{nm}) + 60\,\text{MHz} \end{aligned} \tag{3.12}$$

To calculate the final frequencies, we must account for all of the frequency shifting optics the beams encounter before arriving at the atoms. For the $F=4$ beam we have

$$\begin{aligned} f_{F=4} &= f_{\mathsf{Coh}} - f_{\text{shAOM}} - f_{\text{swAOM}} + f_{\text{sy4AOM}} \\ &\simeq f(F=4 \rightarrow 3') + 60\,\text{MHz} - 40\,\text{MHz} - 120\,\text{MHz} + 100\,\text{MHz} \\ &\simeq f(F=4 \rightarrow 3') \end{aligned} \tag{3.13}$$

where $f_{\mathrm{sy4AOM}} = f_{\mathrm{sy4near}} = f_{\mathrm{sy4far}}$ represents the frequency of either $F=4$ switchyard AOM. Although the rf signal to the near and far AOMs is controlled with different rf switches, both signals originate from the same frequency source. Similarly for the $F=3$ beam,

$$f_{F=3} = f_{\mathsf{Coh}} + n f_{\mathrm{EOM}} + f_{\mathrm{shAOM}} - f_{\mathrm{swAOM}} - f_{\mathrm{sy3AOM}}$$

where $n = 0, \pm 1, \pm 2, \ldots$ distinguishes which modulation order the Fabry-Perot filter cavity is locked to. Since the $F=3 \rightarrow F=3'$ transition has a higher frequency than the $F=4 \rightarrow F=3'$, we always lock to the $n = +1$ order.

$$\begin{aligned} &\simeq f(F=4 \rightarrow 3') + 60\,\mathrm{MHz} + f_{34} + 100\,\mathrm{MHz} + 40\,\mathrm{MHz} - 120\,\mathrm{MHz} - 80\,\mathrm{MHz} \\ &\simeq f(F=4 \rightarrow 3') + f_{34} \\ &\simeq f(F=3 \rightarrow 3') \end{aligned} \tag{3.14}$$

3.3.3 Tracer laser

Unfortunately, although the $F=3$ and $F=4$ are phaselocked to each other at the microwave beatnote, once they split into different directions at CP2, they each encounter different optics before arriving at the vacuum chamber. If any of these optics move or if the effective index of refraction through the different fibers changes slightly due to environmental temperature or mechanical changes, then the two Raman beams will no longer be in phase when they reach the atoms. To correct for all of the relative phase accumulated after CP2, another laser called the "tracer" laser is overlapped with the Raman beams at CP2. It exactly copropagates with the Raman beams until the top of the chamber where the upward going tracer beam is combined with the downward going beam at a photodiode. The part of the tracer beam that traverses the switchyard overlapped with the $F=3$ beam is shifted by one of the $F=3$ switchyard AOMs (near or far) by a frequency of $-f_{\mathrm{sy3AOM}} = -80\,\mathrm{MHz}$. Similarly, the opposing part of the tracer beam that traverses the switchyard overlapped with the $F=4$ beam is shifted by one of the $F=4$ AOMs by a frequency of $+f_{\mathrm{sy4AOM}} = +100$ MHz. Thus, the upward going and downward going tracer beams at the vacuum chamber always have phase difference $2\pi(-180\,\mathrm{MHz})t + \phi(t)$, where $\phi(t)$ represents any phase noise due to the relative motion of optical elements accumulated after the beams separate at CP2. A model S2381 avalanche photodiode from *Hammamatsu* detects this 180 MHz beatnote. The tracer phaselock loop (PLL) compares this beatnote with a 180 MHz reference traceable to the LORAN C 10 MHz reference and controls a 100 MHz VCO. The 100 MHz VCO generates the frequency for the $F\!=\!4$ switchyard AOMs at frequency f_{sy4AOM}. Because the $F\!=\!4$ switchyard AOMs control the frequency and thus the phase of the tracer and Raman beams, this feedback loop effectively removes $\phi(t)$, the relative phase accumulated after the beams split at CP2.

The tracer laser comes from a laser diode that was custom anti-reflection coated by *New Focus*. It is mounted similarly to the other laser diodes in this experiment (see Figure 3.8) and is passively frequency stabilized by a grating in the Littrow configuration to a wavelength of 896.68 nm. As shown in Figure 3.8, the output beam passes through two elliptical lenses which correct its asymmetry, a single-stage optical isolator, and finally a $350\,\mu$m diameter pinhole to improve the beam quality. We typically achieve a transmission efficiency of $\sim 50\%$ and an output power after the pinhole of $\sim$3 mW. Using two lenses, this single-frequency cw laser beam is mode matched to the Raman beams in the switchyard. Finally, two mirrors direct the beam into the other input port of the calcite polarizer CP2. The single-stage optical isolator rotates the polarization by $\sim 45°$, so CP2 splits the tracer beam into roughly equal parts. Note that because the tracer beam enters the other input port of CP2, on each output path the tracer beam emerges polarized orthogonal to the Raman beams. The beams remain orthogonally polarized through the top fiber and to the top PBS cube where the tracer is deflected toward the photodiode instead of continuing on toward the vacuum chamber along with the top Raman beam. At the bottom PBS cube, on the other hand, if the tracer and Raman beam are orthogonally polarized, then the tracer would be deflected away and not pass through the chamber where it can be directed onto the photodiode by the top PBS cube. To convert the beams from orthogonally polarized to parallel polarized, we insert a 20 mm thick piece of quartz crystal before the input to the bottom fiber. This quartz crystal was cut, polished, and AR coated by *TwinStar Optics* to act as a very high order waveplate.

Incident light polarized along the crystal's optical axis experiences a different index of refraction than does light incident along the orthogonal polarization. For the Raman beams at 894.60 nm this index of refraction difference is almost 10 wavelengths. The tracer beam detuned by only +2 nm experiences almost the same shift except for a difference of approximately $\lambda/2$. This thickness of quartz crystal thus acts as a λ-plate for one wavelength and a $\lambda/2$-plate for the other wavelength. By varying the angle of incidence to the crystal, we vary its effective thickness and thus can install it so that it leaves the Raman beam untouched but rotates the tracer beam polarization by $\sim 90°$. The Raman and tracer beams thus emerge from the bottom fiber polarized in the same direction. Since their polarizations are parallel, they both pass undeflected through the bottom PBS cube, through the bottom $\lambda/4$-plate, through the vacuum chamber, through the top $\lambda/4$-plate and into the top PBS cube. Since the top and bottom $\lambda/4$-plates are aligned so that the top and the bottom beams both have the same circular polarization inside the vacuum chamber, light that passes through both the bottom and then the top waveplates will emerge orthogonally polarized and thus be deflected by top PBS cube.

The top PBS cube is used to combine the upward going and downward going tracer beams at the photodiode (see Figure 3.14). Because the top PBS cube deflects the top tracer beam away from the photodiode, we install a zero-order $\lambda/4$-plate and a retro-reflecting mirror on the side of the PBS cube opposite the photodiode. The $\lambda/4$-plate which is double-passed by the top beam rotates the polarization of the top beam by 90° so that the beam will not be deflected twice by the PBS cube. Instead, it will pass horizontally through the cube and overlap with the bottom beam, which is already deflected toward the photodiode. A 10 cm focal length lens collects the light and focuses it onto the 100 μm square active area of the photodiode.

Note that because the upward going bottom tracer and Raman beams have the same polarization, the bottom Raman beam will also be deflected by the top PBS cube. Since the Raman beam is ~ 100 times more intense than the tracer beam, the presence of the bottom Raman beam can cause problems with the photodiode signal. First, it tends to saturate the photodiode which reduces the contrast and thus the signal-to-noise ratio of the 180 MHz beatnote. Second, when the bottom

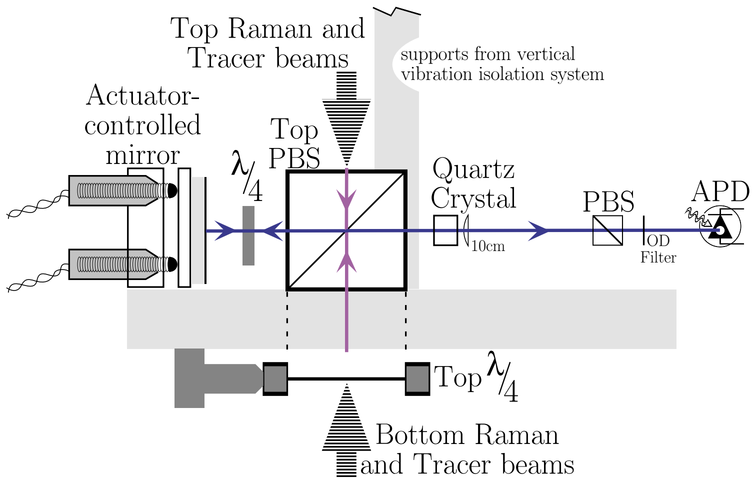


Figure 3.14: **Interferometer platform** used to detect the beat between the upward and downward propagating tracer beams. In order to correct for the phase noise from the relative motion of all of the optics after the Raman beams separate into the switchyard (see Figure 3.10), we use the top 2 inch polarizing beam splitter (PBS) to overlap the top and bottom tracer beams. The bottom tracer beam reflects directly into the avalanche photodiode (APD). The top tracer beam, however, requires a $\lambda/4$-plate to rotate its polarization by 90° and a mirror to redirect it back toward the photodiode. A quartz crystal that acts like a very high order waveplate plus a 0.5 inch PBS are used to filter out the bottom Raman light that is reflected by the top PBS. Phase lock loop (PLL) electronics (see Appendix B) compare the 180 MHz beat signal with a reference signal and control a 100 MHz voltage-controlled oscillator (VCO). This VCO drives the $F=4$ switchyard AOMs (near and far) and thus affects the difference frequency of both the Raman and tracer beams. With the feedback loop closed, all of the phase noise due to the relative motion of the optics is removed and replaced by that due to the motion of the top PBS. In order to reduce the motion of this single optical element, the interferometer platform that holds the top PBS is suspended from the active vertical vibration isolation system described in Section 3.4.

Raman beam turns on or off rapidly, it causes a transient but repeatable glitch in the error signal of the tracer PLL. To minimize these problems, we attempt to use the wavelength difference between the tracer and Raman beam to filter out the Raman light while leaving the tracer alone. In the current tracer setup, we use another quartz crystal, identical to the one before the bottom fiber, and a PBS cube. As with the crystal before the bottom fiber, this piece of quartz is aligned so that it leaves the Raman beam alone but acts like a $\lambda/2$-plate for the tracer. The polarizations of both the top and bottom tracer beams rotate by $\sim 45°$. The polarizer is then set to deflect the Raman beam which emerges from the top PBS cube P-polarized and pass the orthogonal S-polarization. The projection of both the top and the bottom tracer beams onto this polarization passes through the polarizer and interferes at the photodiode. Note that because the top Raman beam is oppositely polarized, this filter does not work for this beam. However, because the tracer and Raman light emerge from the top fiber orthogonally polarized, very little of the top Raman beam makes it through the top PBS cube, so this Raman beam does not need to be filtered out.

In addition to this final configuration of the tracer beam path, much of our data set was taken with the tracer laser in a slightly different setup. Originally, instead of overlapping the tracer beam with the Raman beams at CP2, the beams were combined using an uncoated glass plate tilted at an angle so that it reflected roughly 10% of the light incident on each surface. It thus transmitted 80% of the Raman light and reflected 10% of the tracer light. Because this tracer beam entered the same port of CP2 as the Raman beams did, it emerged polarized in the same direction. Thus, the bottom beam which requires parallel tracer and Raman polarization did not require the additional crystal quartz optic. The top beam, however, always requires that the tracer and Raman beam be orthogonally polarized so that the tracer beam and not the Raman beam will be deflected toward the photodiode. To rotate the relative polarizations of the tracer and Raman beams, the quartz crystal was placed before the top fiber instead of before the bottom fiber. At that time the tracer laser was set to a wavelength of 886.6 nm, or roughly 8 nm *lower* than the Raman beams. Because the wavelength separation was four times larger than the +2 nm separation it has now, we used a different quartz crystal that was roughly four times thinner than the 20 mm one we use now. The layout of the tracer beam was changed to increase the amount of Raman and tracer light emerging from the fibers. The tracer wavelength was changed in the hope that with a smaller wavelength difference between it and the Raman beams, it would better correct the phase error due to mirror motion. To understand how this correction depends on the wavelength difference $\Delta\lambda = \lambda_{\text{tracer}} - \lambda_{\text{Raman}}$, consider a hypothetical mirror that the tracer and Raman beams reflect off at normal incidence. If the mirror moves by Δz toward the incoming beams, the outgoing beam will be shifted in phase by $2k_{\text{Raman}}\Delta z = 4\pi\Delta z/\lambda_{\text{Raman}}$ relative to its input phase. Similarly, the tracer beam will experience the phase shift

$4\pi\Delta z/\lambda_{\text{tracer}}$. This is the phase shift that is detected by the photodiode and used by the tracer PLL to correct both beams. Since this correction differs from the phase shift the Raman beam experienced, the net phase error on the Raman beam will be $4\pi\Delta z(1/\lambda_{\text{Raman}} - 1/\lambda_{\text{tracer}}) \simeq 4\pi\Delta z(\Delta\lambda/\lambda^2)$, proportional to $\Delta\lambda$. After making these two fundamental changes, we found that the improved signal-to-noise of the beatnote and the smaller wavelength difference neither helped nor hurt the signal-to-noise ratio of the final signal.

3.4 Vibration isolation

It is important to note that because the laser fields represent the absolute reference with which we meter the evolution of the phase of the atomic wavefunction, it is absolutely vital that the laser wavefronts be phase stable in both time and space between the first and second and between the third and fourth $\pi/2$-pulses. Thus, with respect to freely falling atoms accelerating due only to the force of the Earth's gravity, the laser wavefronts must not move in space. As discussed in Section 3.6, we compensate for the effect of gravity by changing the difference frequency of the two Raman lasers so that at the center of each pulse the lasers will be resonant with the atoms. In addition to this unavoidable motion due to gravity, the wavefronts also move in time if any of the mirrors off which the beams reflect or if even the laser source itself moves along the beam. As described in Section 3.3.3, a tracer laser overlapped with the Raman beams detects the relative phase change of the two lasers and controls one of the switchyard frequencies to correct the optical phase for any error arising from the motion of the optics. However, because the Raman beams do not reflect off the top polarizing beam splitter (PBS) cube whereas the tracer beam does, if the top PBS cube moves, the tracer phaselock electronics will detect and inappropriately correct for this additional phase. The tracer phaselock feedback loop thus removes the effects of the motion of all of the other optics but then adds a phase due to the motion of the PBS cube. As we will discuss in more detail below, for fluctuations in the position of the PBS cube, the interferometers act like a high-pass filter with a corner frequency of approximately $1/T$, where T is the free-evolution

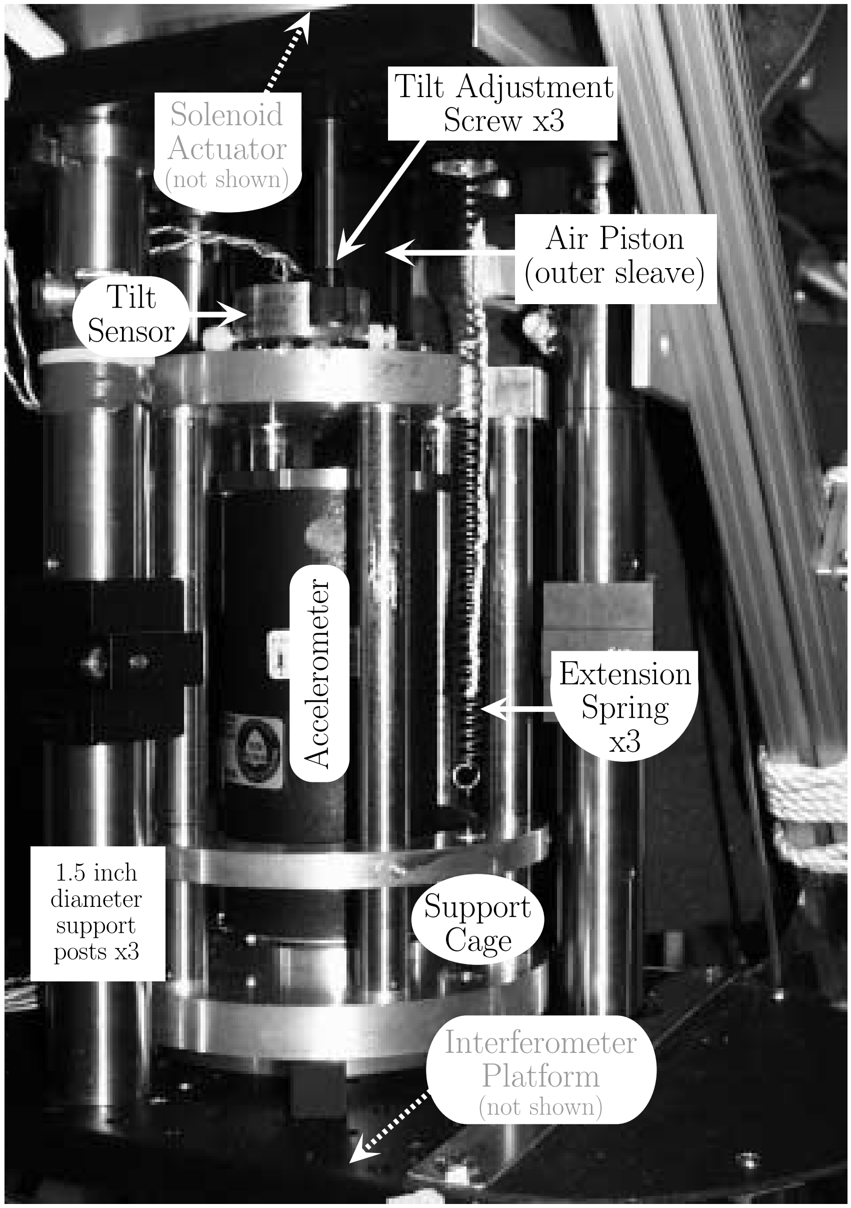


Figure 3.15: **Active vibration isolation system.** An accelerometer is suspended against gravity by three extension springs. The sensor is mounted inside a support cage from which hangs the interferometer platform (not shown) we wish to isolate from environmental vibrations. An air piston constrains the cage from tilting while allowing nearly frictionless motion in the vertical direction. A tilt sensor and adjustment screws allow the orientation of the air piston's outer sleeve to be aligned to be vertical. In real time, a computer system digitizes the sensor's output, transforms it numerically, and outputs the resulting control signal which drives a solenoid actuator (not shown). The actuator controls the entire freely moving column in order to counteract any motion sensed by the accelerometer. The freely moving components are labeled with round bubbles, while the components fixed to the support structure are labeled with square boxes.

time between the $\pi/2$-pulses. Consequently, for large T where we have more resolution, we are also more sensitive to the integrated position noise of the PBS cube, which enters the measurement as phase noise on the fringes and hinders our ability to resolve the phase shift from the photon recoils. To reduce the motion of the PBS cube, we have designed and implemented a single-axis active vibration isolation (VI) system [46].

The VI system shown in Figure 3.15 combines mechanical springs with an electronic feedback loop to produce an almost critically damped spring-mass system with an effective resonance frequency of 0.033 Hz, which significantly reduces the amplitude of vibrations at frequencies from 0.1 to 100 Hz. Attached rigidly to the support platform which holds the top PBS cube and the detection optics for the tracer beatnote (see Figure 3.14) is a column which is suspended by three stainless steel helical extension springs. In addition to the interferometer platform, this column consists of an accelerometer, the inner cylinder of an air piston, and the electrical coil of a solenoid actuator. The air piston made by *Nelson Air* is composed of an outer sleeve and an inner cylinder separated by a tiny air gap. The outer sleeve is attached to the rigid support structure. The inner cylinder is hollow and has two rows of six tiny holes placed symmetrically around its circumference. The holes allow air to escape so that when the inner cylinder is pressurized with air, a thin layer of air flows between the cylinder and the outer sleeve. The pressure of this layer of air constrains the inner cylinder from tilting while at the same time allowing nearly frictionless motion along the direction defined by the outer sleeve.

The *Guralp Systems* CMG-3V accelerometer measures the acceleration of the entire column. A computer digitizes the acceleration signal using a model AT-MIO-16XE-50 data acquisition board from *National Instruments* and processes it internally to produce the control function shown in Figure 3.16d. Besides ease of optimization, digital implementation is advantageous because it can minimize low frequency electronic drift. The 16-bit A-to-D converter is programmed to sample at 4 kHz so that the computer can perform a 4-point running average, thus producing an effective sample rate of 1 kHz, which is 10 times faster than the fastest transfer function parameter implemented. Because of the high-order low-pass filters internal to the sensor, there is virtually no signal at frequencies above 1 kHz and thus very little aliasing noise. The output of this digital control function is then converted back to an analog signal which drives a voltage-to-current converter whose output controls the *BEI Motion Systems* LA12-12A solenoid actuator.

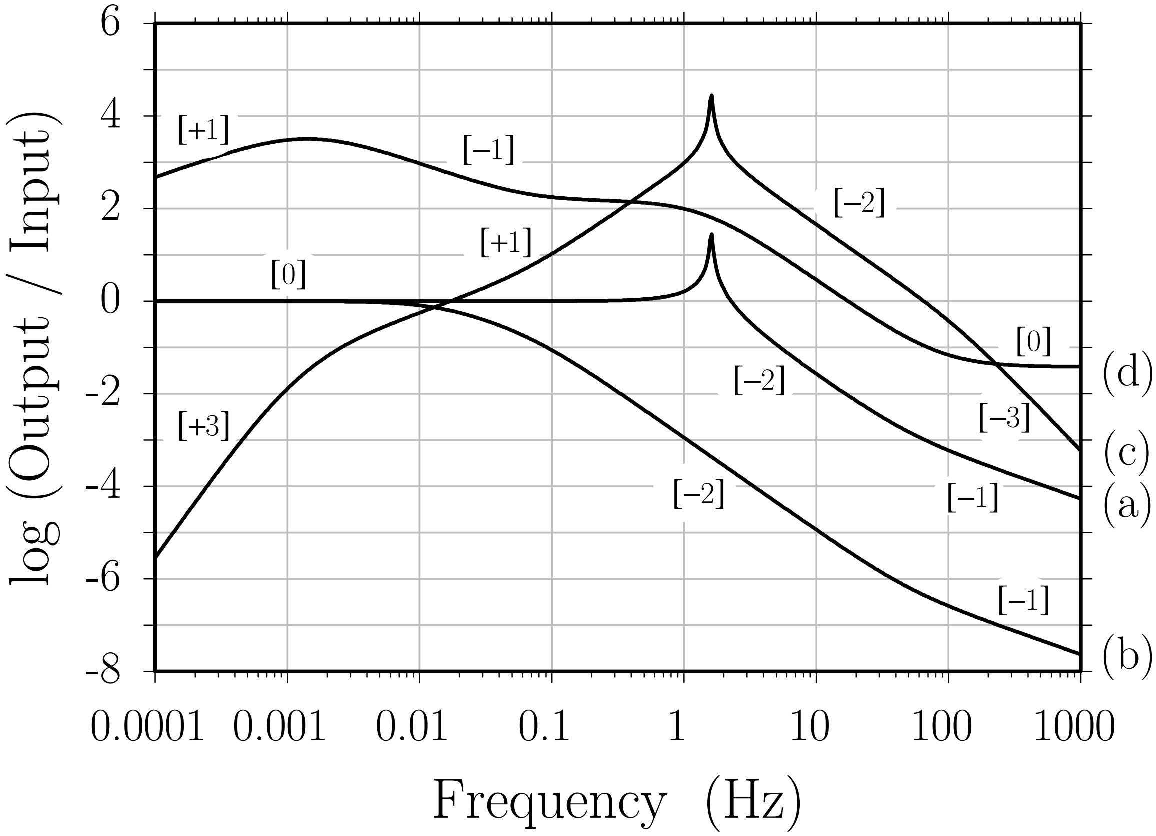


Figure 3.16: **Theoretical transfer functions.** (a) and (b) represent the motion of the mass divided by the motion of the ground for: (a) a spring-mass system with a natural resonance frequency of $f_0 = \omega/(2\pi) = 1.6$ Hz and very little damping ($\zeta_0 = 0.017$), and (b) the same spring-mass after applying two driving forces, $F_{\mathsf{a}} = -mG\ddot{z}$ and $F_{\mathsf{v}} = -2m\omega_0 H\dot{z}$, which lower the effective resonance frequency and thus increase the range over which the system tends to isolate motions of the ground from motions of the mass. Notice from the predicted closed-loop gain (b) that ground motion faster than $\sim$0.1 Hz is reduced when coupling into motion of the mass. (c) shows the implemented open-loop gain including the feedback electronics, the measured response of the solenoid actuator, the spring-mass system, and the finite bandwidth of the accelerometer. (d) is the predicted gain of the feedback electronics alone which provides the F_{v} force, with $H = 46$. The overall gain including the conversion factor of 15.5 for the solenoid actuator provides the F_{a} term with $G = 2\,300$. The numbers in brackets [ ] represent the local slope of different sections of the curves (1st order, 2nd order, . . .)

With the active feedback disabled, the motion $z(t)$ of the column is passively decoupled from the motion $z_g(t)$ of the ground by the mechanical springs. Assume the total mass of the column is m and that the effect of the three springs can be described by a single linear spring constant k and damping factor β. This system has equation of motion

$$\ddot{z} + 2\zeta_0\omega_0(\dot{z} - \dot{z}_g) + \omega_0^2(z - z_g) = 0 \tag{3.15}$$

where $\omega_0^2 = k/m$ is the natural resonance frequency and $\zeta_0 = \beta/(2m\omega_0)$ is the natural damping constant, with $\zeta_0 = 0$ implying no damping and $\zeta_0 = 1$ for a critically damped system. We assume the system is linear and decompose the ground motion into Fourier components $z_g(t) = \tilde{z}_g(\omega)\exp(i\omega t) + \text{c.c.}$ The steady-state solution then has the form $z(t) = \tilde{z}(\omega)\exp(i\omega t) + \text{c.c}$, where $\tilde{z}(\omega)$ is in general complex to represent the phase difference between motion of the ground and the motion of the mass. Inserting these solutions into equation (3.15) gives the frequency domain transfer function

$$\frac{\tilde{z}(w)}{\tilde{z}_g(w)} = \frac{\dot{\tilde{z}}(w)}{\dot{\tilde{z}}_g(w)} = \frac{\ddot{\tilde{z}}(w)}{\ddot{\tilde{z}}_g(w)} = \frac{2(i\omega)\zeta_0\omega_0 + \omega_0^2}{-\omega^2 + 2(i\omega)\zeta_0\omega_0 + \omega_0^2} \tag{3.16}$$

which is plotted in Figure 3.16a for a typical mechanical spring with natural resonance frequency of 1.6 Hz and very little damping ($\zeta_0 = 0.017$). For frequencies greater than $\omega' = \omega_0\sqrt{1 - \zeta_0^2}$, there is reduced coupling between the ground and the mass. For $\omega < \omega'$, any motion of the ground couples directly into motion of the mass. Furthermore, when the natural damping constant is small, any motion of the ground at $\omega \sim \omega'$ is amplified when transmitted to the mass. For $\omega > \omega'$, the response in the frequency domain falls off as $1/\omega^2$ and, because of the damping, as $1/\omega$ for $\omega \gg \omega'$.

To further isolate the mass from ground motions, the active system adds two driving force terms, F_a and F_v, to the right-hand side of equation (3.15). $F_a = -mG\ddot{z}$ is proportional to the mass' acceleration and lowers the resonance frequency but also the damping. To compensate for this loss of damping, we add an additional term $F_v = -2m\omega_0 H\dot{z}$ proportional to the mass' velocity. Since the input to the feedback loop is a voltage proportional to acceleration, the F_a and F_v terms are implemented as proportional gain and integral gain (proportional to $1/\omega$ in the frequency domain),

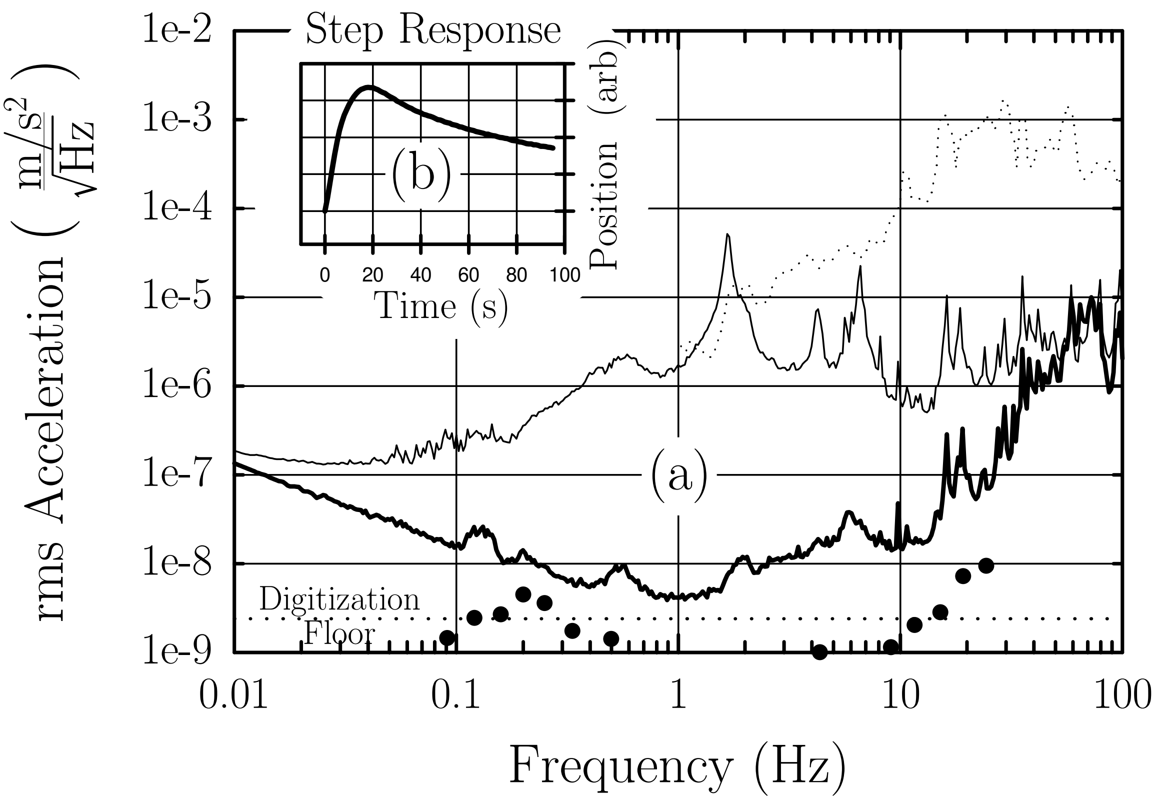


Figure 3.17: **Reduction of the vibrational error signal.** The logarithmic vertical axis of (a) shows equivalent acceleration noise measured at the output of the sensor. The solid and dotted spectra were taken with no feedback with the accelerometer suspended from the springs (solid) and resting on the floor of the lab (dotted). Comparing these two spectra reveals the noise reduction from the optical table and the mechanical springs. The bold spectrum is the error signal with the feedback loop closed. The dashed line shows the theoretical quantization noise limit for a 16-bit A-to-D converter sampling at 4 kHz. The solid points represent the specified noise level of the accelerometer. The position response to a step in the current to the solenoid actuator shown in (b) indicates an almost critically damped system.

respectively. Equation (3.16) becomes

$$\frac{\ddot{z}(w)}{\ddot{z}_g(w)} = \frac{2(i\omega)\zeta_1\omega_1 + \omega_1^2}{-\omega^2 + 2(i\omega)\zeta_1\omega_1 + \omega_1^2} \tag{3.17}$$

where $\omega_1 = \omega_0/\sqrt{G+1}$, $\zeta_1 = (\zeta_0 + H)/\sqrt{G+1}$, and G and H are constants that will be controlled by the feedback electronics. Figure 3.16b shows this predicted closed-loop response with reduced resonance frequency and increased damping for the values actually implemented, $G = 2\,300$ and $H = 46$. Figure 3.16c shows the overall open-loop gain. Figure 3.16d shows the predicted gain of the feedback electronics alone. Notice that in addition to F_a and F_v, this transfer function includes several other terms which are required for stability because of the finite bandwidth of the sensor.

By inputting a step in the current to the solenoid actuator, we observed the step-response of the closed-loop system (Figure 3.17b). By observing this step-response for successively smaller damping constants, we were able to fit the response to the theoretical expression for a second order system and obtain a value for the new effective resonance frequency, $f_1 = \omega_1/(2\pi) = 0.033$ Hz.

Figure 3.17a shows that the vibration isolation system plus the optical table reduce the acceleration error signal of the sensor by as much as 1000 from 0.01 to 100 Hz. By comparing the spectrum taken on the floor of the lab (dotted line) with the spectrum taken on the vibration isolation tower when the accelerometer is attached rigidly to the supporting tower (not shown), we observe that the optical table reduces the noise from 10 to 100 Hz by $\sim$10 and amplifies the noise around 3 Hz by a factor of $\sim$10. With the accelerometer freely swinging (solid line), the mechanical springs provide most of the high frequency isolation by reducing the noise above 10 Hz by another factor of 100 and somewhat reducing the noise due to the resonance of the optical table. The large peak in this spectrum at 1.6 Hz is due to the resonance of the mechanical springs. Notice that the springs do not affect the acceleration noise below 1 Hz. Only with the feedback loop closed (bold line) does the error signal from 0.01 to 1 Hz become smaller by as much as a factor of 300. The features at 0.14 Hz and around 0.55 Hz most likely represent seismic motion of the ground and wobbling of the building, respectively.

Without another independent sensor we cannot measure the true noise floor of the isolation system. However, as a practical demonstration that the system works, Figure 3.18 shows the reduction of interferometer phase uncertainty for different fringe periods. For our interferometers in the limit that $T' \to 0$, acceleration of the laser wavefronts with Fourier component $\tilde{a}(\omega)\cos(\omega t)$ causes a phase error

$$-2\pi\frac{\tilde{a}(\omega)}{\lambda\omega^2}2\sqrt{2}\left|\sin\left(\frac{\omega T_{\rm rep}}{2}\right)\right|\;\sin^2\left(\frac{\omega T}{2}\right) \tag{3.18}$$

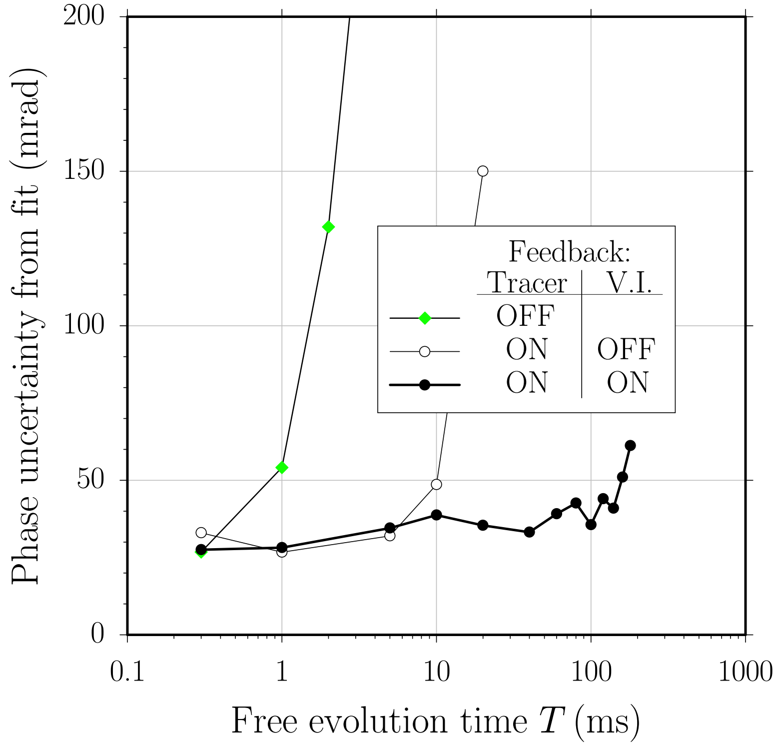


Figure 3.18: **Vibration isolation performance for atom interferometry.** The vertical axis shows the uncertainty in the phase when interferometer data are fit by a sine wave. The tracer phaselock feedback loop corrects the phase error due to any relative change in the optical path length traveled by the two tracer beams. Because the interferometer beams do not reflect off the top polarizing beamsplitter cube, the tracer feedback loop removes all of the phase noise on the interferometer beams up to the cube but then inappropriately adds phase noise due to the motion of the cube. With the tracer feedback loop inactive (gray diamonds), none of the phase error due to the relative motion of the optics is canceled and interferometer fringes are apparent only for the very smallest values of the free evolution time T. With the tracer feedback loop active (hollow circles), the phase noise of the fringes is then limited by the motion of the beamsplitter cube which is *passively* isolated from ground motion by the optical table and the mechanical springs of the vibration isolation (VI) system. The improvement of the *active* isolation (solid circles) is evident. Without active isolation, fringes with period greater than 30 ms are completely washed out. Since the sensitivity of our measurement increases linearly with fringe period, it is clear that these types of precision interferometer measurements would have been impossible without active vibration isolation.

where λ is the light wavelength, T is the interferometer fringe period, and T_{rep} is the launch repetition period. Note that for acceleration with frequencies $\omega > \pi/T$, the measurement acts as a second-order low-pass filter. More importantly, for $\omega < \pi/T$ this sensitivity to wavefront motion increases proportional to T^2. For long fringe periods ($T \simeq 160$ ms) where we have the most sensitivity, the interferometer is $160^2 = 25\,600$ times more sensitive to accelerations than it is when $T \simeq 1$ ms. Thus, without vibration isolation, even small background vibrations would completely wash out these long-period fringes.

3.5 Magnetic fields

The energies of the cesium hyperfine ground states in an external field B are given by the Breit-Rabi equation

$$E(F, m_F) = -\frac{hf_{34}}{2(2I+1)} - m_F g_I \mu_B B \pm \frac{hf_{34}}{2}\sqrt{1 + \frac{4m_F}{2I+1}x + x^2} \tag{3.19}$$

where $f_{34} = 9\,192\,631\,770$ Hz is the defined hyperfine splitting, $I = 7/2$ is the nuclear spin, $\mu_B = 1.400$ MHz/G is the Bohr magneton, and

$$x = \frac{(g_J + g_I)\mu_B B}{hf_{34}} \tag{3.20}$$

The sign depends on the relative orientation of the electron's spin: $(-)$ for the $F{=}3$ state and $(+)$ for the $F{=}4$ state. For cesium, $g_I = 0.000\,398\,853$ and $g_J = 2.002\,540$ [47]. The term in equation (3.19) proportional to x is the linear Zeeman shift. Since its sign is opposite for the $F{=}3$ and $F{=}4$ states, for a magnetic field B (expressed in mG) the hyperfine splitting for m_F conserving transitions changes by

$$\begin{aligned}\Delta(f_4 - f_3) &\simeq \frac{2(g_J + g_I)}{2I+1}\mu_B m_F B \\ &= \frac{2(2.002\,939)}{8}(1.400\,\text{kHz/mG})\,m_F B \\ &= (0.7008\,\text{kHz/mG})\,m_F B \end{aligned} \tag{3.21}$$

Because it does not shift the $m_F{=}0$ sublevels, we would like to build the interferometers using only these states. By performing Doppler-free (DF) two-photon transitions between individual magnetic sublevels (see Section 3.1.6 discussing the DF Raman laser), we are able to selectively detect only the atoms in the magnetic field insensitve $m_F = 0$ sublevels. This selective detection works only as long as the magnetic sublevels are not all degenerate. Thus, during the DF Raman transition, all the magnetic field sensitive levels ($m_F \neq 0$) must be shifted by at least the spectral width of a DF Raman π-pulse. Additionally, a magnetic bias field is required for the interferometers. If the spectral width of the Doppler-*sensitive* transitions driven by the Raman beams is larger than the shift of the magnetic sublevels, then atoms in these states will be carried through the interferometer. Due to polarization impurities of the Raman beams or a misalignment between the Raman beams and the magnetic bias field, the magnetic field sensitive levels may mix the magnetic field insensitive levels and thereby shift the final phase of the interferometer[7]. At the very least, even if the presence of the other magnetic levels does not change the final phase, they may reduce the interferometer contrast by increasing the phase randomized background. Thus, we require a magnetic bias field sufficiently strong to shift all of the $m_F \neq 0$ Zeeman levels out of resonance with DF Raman transitions with width $\sim$64 Hz and the interferometer $\pi/2$-pulses with halfwidth of $\sim$ 67 kHz. Unfortunately, because the magnetic bias field will never be perfectly uniform, the quadratic Zeeman shift will affect the $m_F = 0$ levels. As discussed in Section 6.4, this effect is proportional to the bias field level. Thus, to minimize potential systematic errors due to magnetic phase shifts, we must minimize the magnetic bias field and its spatial fluctuations. We choose a magnetic bias field of 71.6 mG which according to equation (3.21) shifts the $F = 3, m_F = 1 \rightarrow F = 4, m_F = 1$ transition by 50 kHz as a compromise for these competing goals.

The magnetic bias field is generated by sending electrical current through a solenoid that is wound inside the triple-layer 0.025 inch thick *Hipernom* magnetic shielding (see Figure 3.19). The solenoid consists of 1.0 mm diameter Kapton-insulated magnet wire wound up and continuously back down a 2 inch diameter aluminum tube held coaxially within the magnetic shielding. The effective turn ratio is $\sim$19 turns/cm. Two smaller 8 turn solenoids wound at half this rate are wrapped in the same direction, one on either end of the main bias solenoid. These smaller

[7]See Sections 6.1.5 and 6.4 for a discussion of this systematic effect.

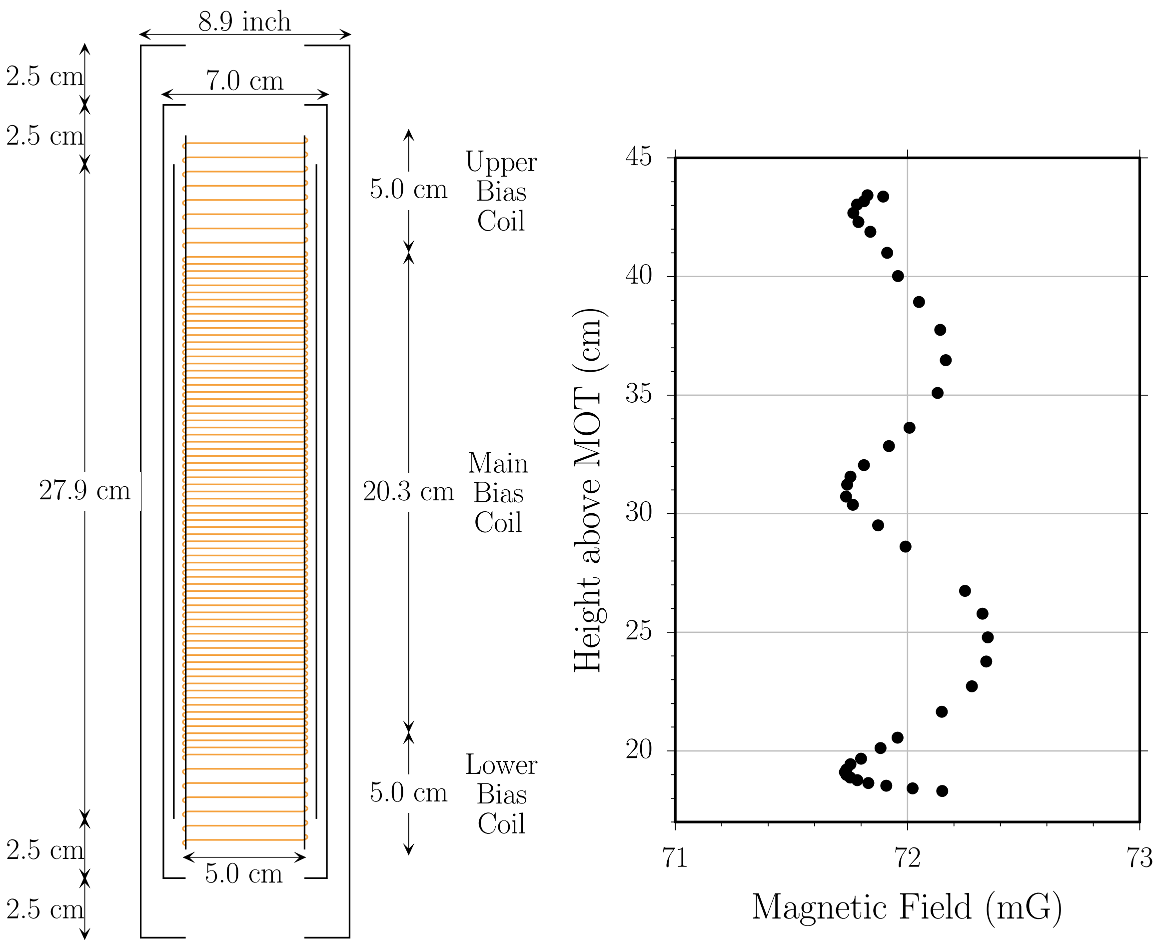


Figure 3.19: **Magnetic field strength inside the magnetic shielding.** Inside the vacuum chamber three layers of cylindrical magnetic shields are placed concentrically around an aluminum tube around which magnetic wire is wound (shown in gray). The shielding reduces the external magnetic field contribution, while the solenoid windings allow us to apply a controllable non-zero bias of approximately 72 mG as shown in the graph on the right, which has the same vertical scale as the figure on the left. The field's spatial variation is due to leakage and end effects of the shielding. To minimize this spatial variation, each end of the solenoid is separated into an independent coil. By setting the current through the "upper" and "lower" bias coils independent of the current through the "main" bias coil, we can compensate for these end effects and make the field strength near the ends the same as the value in the center, effectively doubling the usable range inside the shielding to over 26 cm.

"bias trim coils" can be controlled independently and are used to reduce the spatial variation of the main bias field. With no current running through any of the bias coils, the magnetic field inside the shielding is determined by the ability of the shielding to redirect the external field lines. The external field comes primarily from the MOT trim coils (see Section 3.1.4) which are set to apply a large magnetic field (> 500 mG) in the vertical direction after the atoms are launched. The magnetic shielding should reduce this field by at least four order of magnitude [48], well below the desired field level of around 70 mG. The measured magnetic field in the vertical direction with 3.00 mA sent through the main bias coil by a source precise to better than 1×10^{-5} is shown in Figure 3.19. The floor of this well-like structure is determined by the main bias coil, while the walls are due to the external field from the MOT trim coils leaking through at the ends of the shielding. To extend the usable range within the shielding, we send -7.3 mA through the lower bias trim coil and -21 mA through the upper bias trim coil to produce a local field opposite to the main bias field. These additional fields subtract from the external field and tend to move the walls out. The particular current values we use were chosen to minimize the peak-to-peak variation of the field within as large a region as possible.

3.6 Interferometer pattern generation

The main experiment timing is controlled by a single DG555 pulse generator from *Stanford Research Systems*. Every 908 ms it triggers all of the channels discussed in Section 3.1 required to stop loading atoms into the MOT and to launch them vertically. At some time after this trigger, it triggers the other half of the experiment which controls the adiabatic transfer beams that generate the interferometers. To build a single interferometer we must be able to control the amplitude, direction, and frequency of the two Raman lasers for each of the possibly over 50 pulses which occur during the approximately 300 ms the atoms spend inside the magnetic shielding. In addition, to arrive at a final recoil value we must generate several different interferometer geometries, so we must change these settings in the remaining 600 ms so that by the time the atoms are next launched the new interferometer geometries will be constructed. This last requirement that we change the settings each second turns out to be the most difficult to achieve. For instance, we must control the two-photon difference frequency for each adiabatic transfer pulse so that the Doppler-sensitive driving lasers are in resonance with the atoms. In addition to the shift of

$f_{\rm rec} = 15.006$ kHz from each photon recoil, after time t the resonance condition will change by $-\mathbf{k}_{\rm eff}gt$ due to the acceleration of gravity, where $k_{\rm eff}g = 2\pi(21.908\,\text{MHz/s})$. Thus, the two-photon difference frequency must be set to a different value for each of the adiabatic transfer pulses. We could find no commercial synthesizer that could output the 50 or so different rf frequencies at triggerable times and then change all of the frequency values in only 600 ms. Another example of the technical difficulties associated with generating the interferometer pulses involves the amplitude shaping of individual pulses. Ultimately, these shapes will be stored digitally and converted to an analog voltage by a digital-to-analog converter (DAC). To adequately define fast edges for a 70 μs π-pulse, for example, a reasonable time resolution would be no slower than 1 μs per sample point. Sampling every μs over the whole interferometer which could last as long as 300 ms would require a array of 300 000 points for each channel, or 600 000 points in total to control both the $F=3$ and $F=4$ amplitude patterns. This pattern length is 20 to 40 times longer than the memory sizes available in commercial devices at the time. Also consider the time that would be required to send 0.6 million 2-byte points across current conventional inter-device communication lines (i.e. GPIB). If these data had to be sent each time a different interferometer was selected, we would have to dramatically reduce our data rate.

To overcome these technical challenges, we assembled a mostly custom-made system specifically designed to run this experiment. The frequencies are controlled by the direct digital synthesizer (DDS) described in Section 4.1. This synthesizer can output frequencies from dc up to around 200 MHz as long the frequency is an integer multiple of the minimum step size $f_{\rm clk}/2^{32} = (1\,\text{GHz})/2^{32} = 0.233$ Hz. This 32-bit integer is set by a model AT-DIO-32/F pc-board from *National Instruments* inside the computer. This pc-board allows direct memory access (DMA), so in order to switch the DDS's output frequency between many different values, all the computer must do is set aside an array of 32-bit numbers in its memory, point the DIO-32/F board to this array, and provide the board with an external trigger whenever a new frequency value is required. Whenever the 32-bit control word changes, the DDS outputs the new frequency in less than 50 ns. For this reason and because the DIO-32/F board and the DDS handle what little handshaking is necessary, it takes less than 2 μs to change the frequencies and the computer is always free to do other things.

The amplitudes and direction of the beams are controlled by another pc-board in the computer. This model AWFG/2 pc-board from *Keithley Metrabyte* functions as a dual-channel arbitrary waveform synthesizer. For each channel it can store up to 16 384 16-bit values in its internal memory. When enabled by a gate signal and driven by a sample clock, the board steps through this memory at a maximum rate of 5 points per μs. 12 of its 16 bits are used by a DA converter on the board to generate a voltage from 0 to 5 V. The remaining four bits determine the state of four TTL logic outputs. The dual-channel board thus has two analog and eight logic outputs.

As discussed above, to digitize an entire 300 ms long interferometer with μs resolution would require many more memory points than are available. To avoid this problem, we use an additional synthesizer to gate the AWFG pc-board's sampling on and off. For the high-sensitivity interferometers most of the time between the first and last $\pi/2$-pulse is spent with all of the light off, so the AWFG sampling can be turned off during this time. Thus, the pattern stored in the AWFG memory for an interferometer with $T = 1$ ms and with $T = 160$ ms will look very similar and use roughly the same number of points. This is clearly advantageous because if the timing resolution used to define the interferometer pulses is the same for both long and short interferometers, the pulses for all interferometers will be as close to identical as possible.

The "gate synthesizer" which turns the AWFG sampling on and off is a model DS345 arbitrary waveform generator from *Stanford Research Systems*. Because the total time for an interferometer varies by over an order of magnitude for different possible values of T, the timing resolution of this synthesizer will definitely vary. However, this resolution should not greatly affect the shape of the pattern, since it outputs only a binary waveform with TTL values 0 or 5 V. When the gate synthesizer's output is at 5 V, the clock for the AWFG enables and immediately starts stepping through its stored memory. When the gate synthesizer outputs 0 V, the AWFG clock continues oscillating, but it does not step through memory. Figure 3.20 shows the contents of the AWFG memory for interferometer $\boxed{1}$ with no π-pulses. If the gate signal shown in Figure 3.25 turns the AWFG waveform generation on and off, the

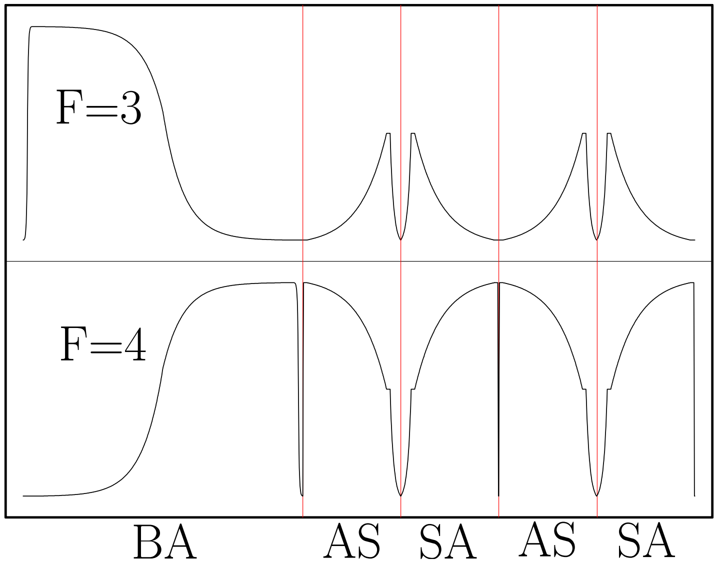


Figure 3.20: **Contents of the AWFG board's two analog channels** for interferometer geometry [1] with no π-pulses. The single velocity selecting π-pulse (BA) and four $\pi/2$-pulses (AS, SA, AS, SA) that build the interferometer are shown. An external gate control switches the sampling on at the beginning and off at the end of each of these pulses. By using this gate control, the memory contents are independent of T, the time between the $\pi/2$-pulses. Because varying T is such an useful investigative tool for finding systematic errors, this feature is an important improvement over early versions of this experiment.

AWFG board will output the pattern shown in Figure 3.26.

Digital output X0 drives the request line to the AT-DIO/32F board and the strobe input of the DDS. The strobe input to the DDS is active low, so that whenever it is held at TTL low voltage, the latch for the 32-bit inputs becomes transparent and the bits are loaded into the frequency control word of the device (see Figure 4.1). As discussed in Section 4.1, as soon as this frequency control word changes, the device will start outputting the new rf frequency determined by the values of the 32 frequency control bits. The AT-DIO/32F board is configured to be active rising, so that when X0 returns from TTL low to TTL high, the next 32-bit frequency value will be transfered from the computer's memory to the digital lines. With X0 held at TTL high, the strobe line to the DDS is disabled, thus preventing the changing frequency control bits from affecting the output rf frequency. In this way, the single AWFG output X0 first tells the DDS to change its output frequency to the value held currently at the frequency control bits by the AT-DIO/32F board. It then tells the

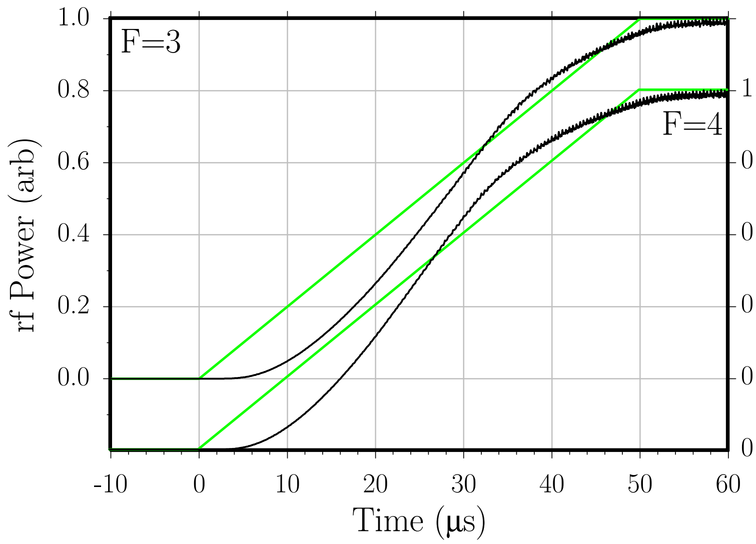


Figure 3.21: **RF power to the shaping AOMs**. A 50 μs long linear ramp (represent in gray) is applied to the control voltage of the two variable radio-frequency (rf) attenuators. The amplified output of these attenuators drives the 40 MHz acousto-optic modulators (AOMs) for the $F=3$ and $F=4$ beams. By setting the rf power to these "shaping AOMs", we control the amount of light diffracted toward the atoms and thereby generate the adiabatic passage pulse shapes. The nonlinear response is due to the variable rf attenuators at low powers and the saturation of the rf amplifiers at high powers. To correct for this nonlinearity, with a fast photodiode we measure the optical power diffracted by both shaping AOMs when a 1 ms long ramp with a 0.4% duty cycle drives the attenuator controls. We then digitize and invert this result using numerical interpolation. Whenever we wish to generate a certain pulse shape, we can apply this inverted transform and remove the nonlinearity to better than 1%.

AT-DIO/32F board to load new values for the next frequency onto the 32 frequency control lines in preparation for the next frequency change.

3.6.1 Pulse shaping

The two analog channels of the AWFG board are used to control the variable rf attenuators (see Figure 3.22) that set the amplitude (and power) of the rf signal applied to the shaping AOMs. The amount of light defracted by the shaping AOMs is proportional to the rf power or the square of the rf amplitude output from the attenuators. For each pulse shape of each channel the computer determines an array of values ranging from 0 to $2^{12}-1 = 4095$ which cause a series of values from 0 to +5 V to be applied to control input of each rf attenuator. Figure 3.21 shows the power of

Table 3.2: **Controls for generating the adiabatic transfer light pulses.**

Name	Source	Description
$F=3$	AWFG:XAn	Adiabatic transfer pulse shapes for $F=3$ light
$F=4$	AWFG:YAn	Adiabatic transfer pulse shapes for $F=4$ light
Com	AWFG:Y3	Switch signal for common switch AOM
SY *Norm*	AWFG:X2	Switchyard *normal* (chirp *up*) or off
SY *Inv*	AWFG:X1	Switchyard *inverted* (chirp *down*) or off
DDS Strobe	AWFG:X0	Trigger DDS to update its output
Chirp *Up* Trig	AWFG:Y2	Trigger frequency sweep of chirp *up* synthesizer
Chirp *Down* Trig	AWFG:Y1	Trigger frequency sweep of chirp *down* synthesizer
DDS	ADS-431	Tunable reference for Raman difference frequency
Gate	DS345	Gate signal for pausing AWFG waveforms
Chirp *Up*	DS345	Linear frequency sweep when switchyard is *normal*
Chirp *Down*	DS345	Linear frequency sweep when switchyard is *inverted*
FF	HP33120A	Feed-forward signal for cavity and cesium locks
CS	HP33120A	Tuning voltage for cesium lock AOM

the rf signal from the attenuators when a $50\,\mu$s long linear ramp is programmed into memory. In order to achieve the desired pulse shape, we invert this curve to determine which light output level is produced by which digital output level. To measure this curve directly, a triangle waveform (rising linear edge followed by falling linear edge) lasting 2 ms was programmed into the AWFG memory. The light emerging from the fibers was measured by a photodiode whose time constant is less than 2 ns. As the amount of rf power to the AOM increases, so does the heat delivered to the piezoelectric actuator. This heat can change the properties of the crystal and cause the AOM to move slightly with respect to the beam, both of which may change the defraction efficiency. To prevent the AOMs from heating up excessively during this measurement, after the triangle waveform we turn the output off for 248 ms before repeating the measurement. This duty cycle of $\sim$0.4% is similar to the effective duty

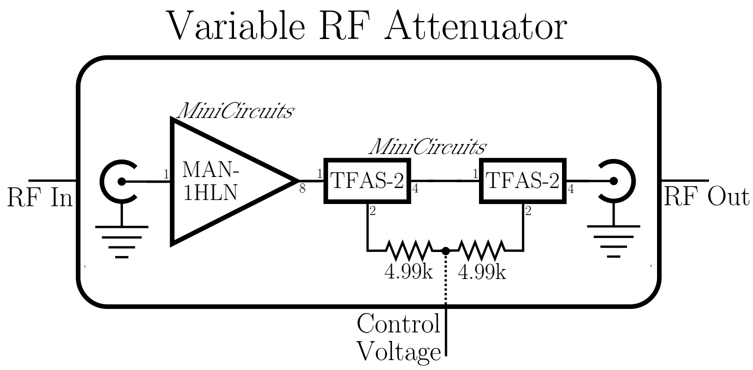


Figure 3.22: **RF attenuator used to generate the adiabatic transfer pulse shapes** by controlling the amplitude of the radio frequency (rf) signal driving the shaping acousto-optic modulators (AOMs). A single supply operational amplifier (not shown) buffers the input control voltage to provide the bias for two *MiniCircuits* TFAS-2 radio frequency (rf) attenuators in series. Internally, these attenuators are rf diodes whose transmission loss depends on the current supplied by the dc bias voltage. For small currents (or bias voltages), this dependence is strongly nonlinear. Because of this non-linearity, we measure the attenuation as a function of control voltage and then modify the shape of the control voltage pulse so that the light intensity diffracted by the shaping AOMs has the desired shape.

cycle of most interferometer patterns, so it should approximate the amount of heating present in the actual experiment. To insure linearity we operate the photodiode at roughly 500 times below its optical saturation point. After approximately 128 averages, we scaled the data and arrived at a curve $I(p)$, where p is a number from 0 to 4095 and $I(p)$ is the light level scaled from 0 to 1. Using numerical interpolation, we inverted this curve to generate the function $p_a(x) = I^{-1}(x)$ where x is the scaled light level with $x = 0$ indicating off and $x = 1$ indicating fully on and $p_a(x)$ is an integer from 0 to 4095. When the computer calculates a desired pulse shape $a(t)$, it programs the waveform $p_a(a(t))$ into the AWFG memory. To test the linearization, after generating the transforms $p_a3(x)$ and $p_a4(x)$ for the two channels, we apply them to the triangle pattern described above and demonstrate that we have linearized the light pulse shapes to better than 1%.

To insure that both the $F\!=\!3$ and $F\!=\!4$ lasers switch on and off together, we installed a third AOM, the common switch AOM. Digital output X3 controls a *MiniCircuits* model ZYSW-2-50DR rf switch which turns on and off the 120 MHz signal to this common switch AOM.

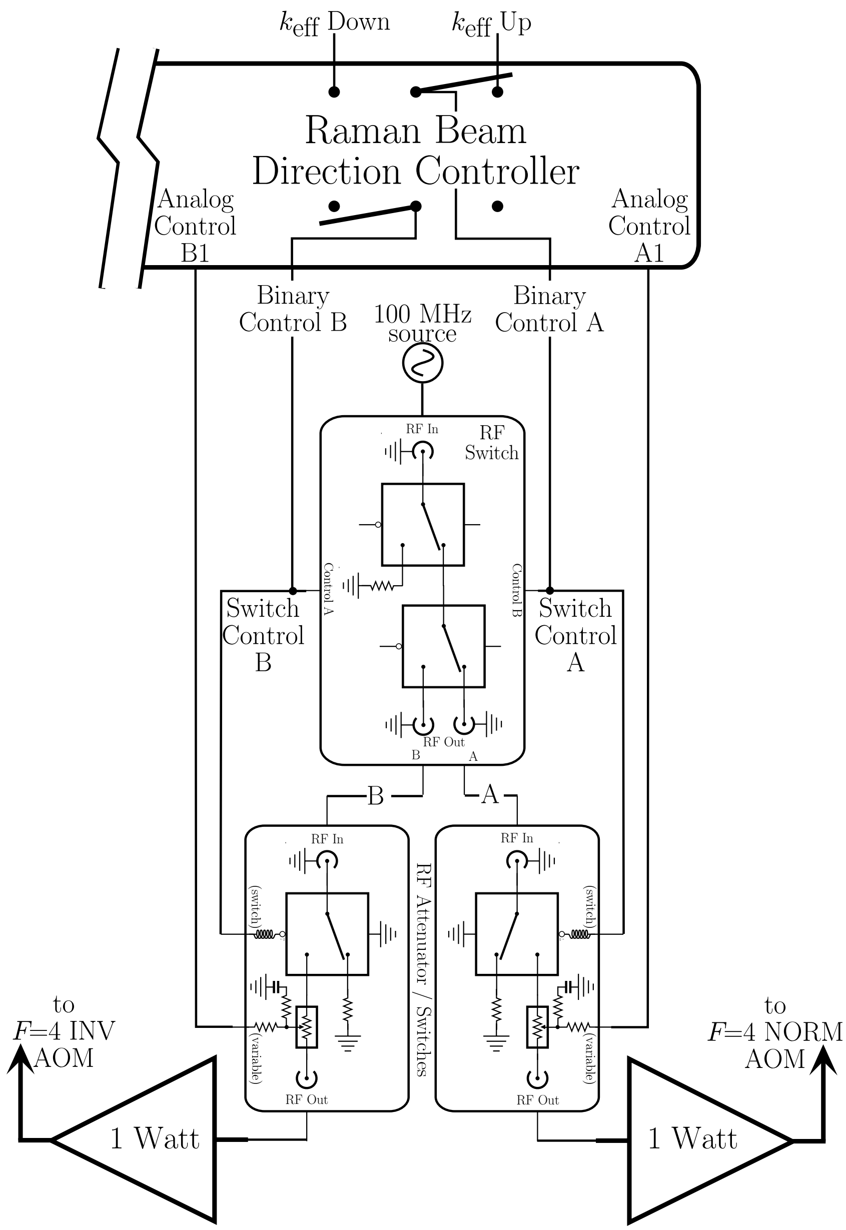


Figure 3.23: The **Raman beam direction controller** uses two logic signals (X1 and X2 from the AWFG board, indicating k_{eff} *Down* and *Up*, respectively) to set the state of the switchyard (*Normal*, *Invert*, or *Off*). It does this by driving a radio frequency (rf) switch and an rf switch/attenuator, which are shown for just the two $F = 4$ switchyard AOMs. The rf switch directs the fixed amplitude 100 MHz source signal to either the normal or inverted AOMs, while the rf switch/attenuator fine tunes the rf amplitude at each switchyard AOM so that the intensities of all possible Raman beam combinations can be matched. To improve the on/off isolation of the switchyard, the rf switch/attenuator also contains an binary switch. Except for the binary controls A and B, the rest of the components in the figure are duplicated for the $F = 3$ switchyard AOMs.

3.6.2 Beam direction switching

Digital outputs X1 and X2 control the k_{eff} down and up electronics, respectively, which together determine the state of the switchyard. These outputs are both connected to the tracer PLL box so that the lock can be put on hold when the switchyard is off and to the "Raman beam direction controller" shown in Figure 3.23 that determines which switchyard AOMs are on. Each of the four switchyard AOMs is driven by an amplified rf signal. The rf signals for the $F=3$ near and far AOMs both come from the same fixed 80 MHz source. Similarly, the signals for the $F=4$ AOMs come from the same variable 100 MHz source. These two rf sources each pass through two *MiniCircuits* model PSW-1211 rf switches. The first rf switch turns the source on or off. The second switch directs the rf signal toward either the near or far AOM. After the directing switch, there are four rf signals, one for each switchyard AOM. These four rf signals each pass through a *MiniCircuits* model TFAS-2 variable rf attenuator and another model PSW-1211 rf switch. The rf switch provides additional isolation to more completely block the rf signal when it is supposed to be off. The four variable rf attenuators are used to balance the beam intensities. All of these components are controlled by the Raman beam direction controller. It takes the two logic outputs X1 and X2 from the AWFG board and converts them to two logic outputs and four analog outputs. The logic outputs of the switchyard control box control the rf switches. The analog outputs drive the variable rf attenuators. For example, if the far AOMs are on, then the binary switches for the two rf sources will both be on, the next switches will be set to direct the rf signals toward the far AOMs, the switch for the $F=3$ and $F=4$ far path will be on, and finally the variable rf attenuators attenuating these two signals will be set to non-zero voltage values determined by the position of two potentiometers on the front of the switchyard control box. These potentiometers provide independent fine control for the light intensity defracted by each particular switchyard AOM. If the switchyard is OFF, all binary switches are off and the variable rf attenuators are set to maximum attenuation. In this case, the rf signal must pass through three binary switches all set to off and one variable rf attenuator set to maximum attenuation before being amplified and sent to the switchyard AOMs. Given this amount of attenuation, the rf amplitude after the amplifiers does not limit the ability of the switchyard to truly turn the light completely off [8].

3.6.3 Frequency chirp during $\pi/2$-pulses

Because the $\pi/2$-pulses are particularly sensitive to the relative phase between the lasers and the atomic wavefunctions, we attempt to maintain the resonance condition throughout these pulses. To counteract the effect of gravity, we must linearly sweep the two-photon difference frequency at a rate of $\partial f/\partial t = \pm g/\lambda_{\text{eff}} = \pm 21.908$ MHz/s, where the sign is negative for $\mathbf{k}_{\text{eff}}$ propagating upward. To accomplish this, we chirp the frequency of the rf signal going to both shaping AOMs. Since the shaping AOMs frequency shift the Raman beams with opposite sign ($+f_{\text{shapingAOM}}$ from the $F\!=\!3$ shaping AOM and $-f_{\text{shapingAOM}}$ from the $F\!=\!4$ shaping AOM), chirping the single frequency $f_{\text{shapingAOM}}$ driving both shaping AOMs at a rate of $\partial f/\partial t$ automatically changes the two-photon difference frequency at a rate of $2\,\partial f/\partial t$. Over the $\pi/2$-pulse pulse length of $250\,\mu$s, the AOM frequency must sweep by only

$$\pm\frac{1}{2}(21.908\,\text{MHz/s})(250\,\mu\text{s}\ ) = 2.74\,\text{kHz}$$

which corresponds to a fractional change in the AOM frequency of

$$(2.74\,\text{kHz})/(40\,\text{MHz}) = 6.85\times 10^{-5}$$

This fractional change is small enough that the change in the resulting changing in the shaping AOMs' diffraction angles should not cause a significant change in the transmission efficiency through the switchyard and the optical fibers.

We use two model DS345 synthesizers from *Stanford Research Systems* to chirp the shaping AOM frequency $f_{\text{shapingAOM}}$, one to chirp up and one to chirp down. Logic outputs Y1 and Y2 trigger the "chirp down" and "chirp up" synthesizers, respectively. The synthesizers each output a cw 20 MHz signal of fixed amplitude. The chirp up synthesizer is programmed to perform a linear sweep in its output frequency from

[8] The isolation of the switchyard seems to be limited by the optical properties of the beams and/or the AOMs. See Table 3.1.

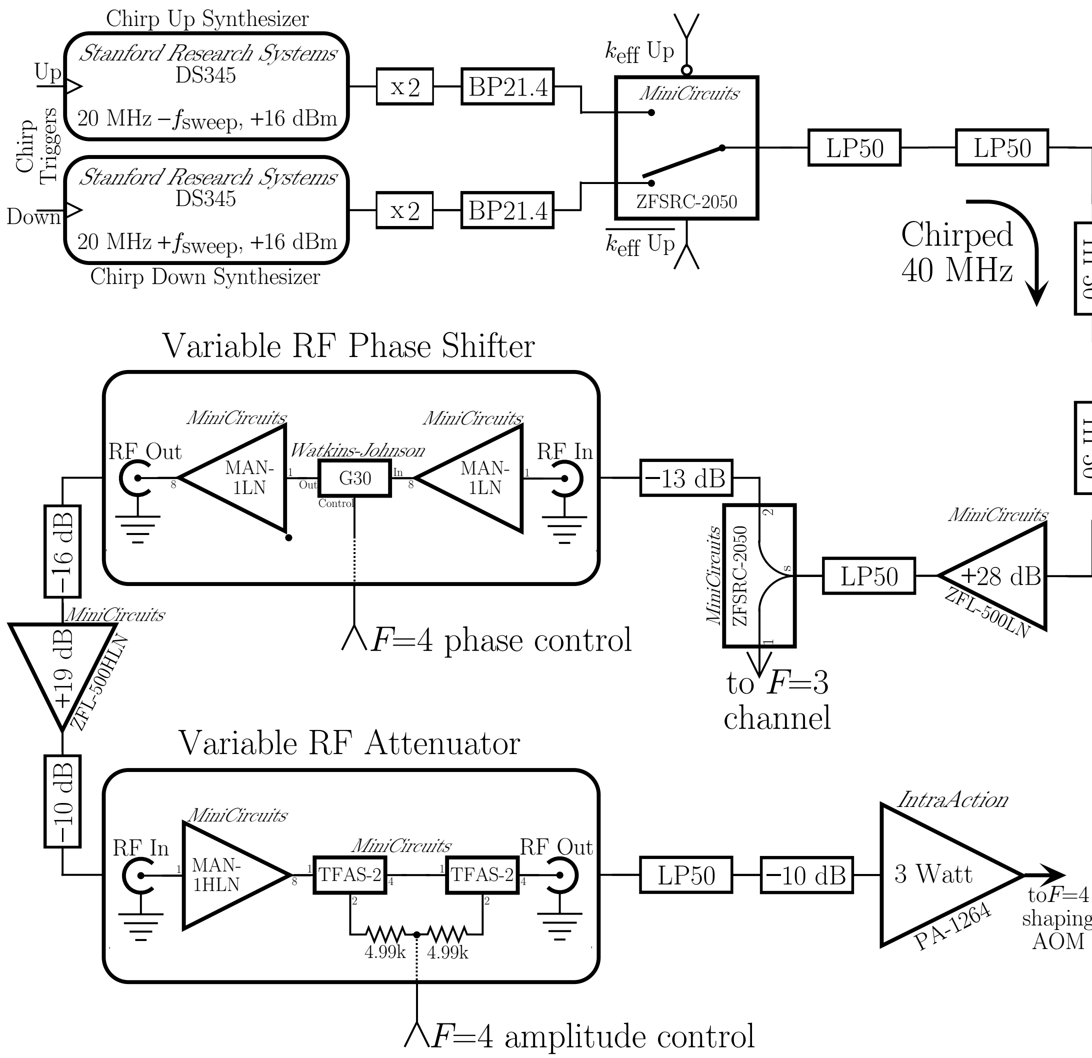


Figure 3.24: **Generation of the chirp signal for the shaping AOMs**. Two *Stanford Research Systems* DS345 arbitrary waveform synthesizers are programmed to linearly sweep their output frequency from frequency 20 MHz $\pm f_{\text{sweep}}$ back to 20 MHz whenever they are externally triggered with a "chirp trigger signal" from the AWFG board. After being frequency doubled, a *MiniCircuits* model ZYSW-2-50DR rf switch selects either the "chirp up" or "chirp down" signal. This rf signal is then filtered and amplified before finally being split by a *MiniCircuits* model ZFSRC-2050 resistive power splitter into two equal parts for the $F=3$ and $F=4$ shaping acousto-optic modulators (AOMs), respectively. To form the adiabatic transfer light pulses, the amplitude of this near 40 MHz rf signal is controlled with a "variable rf attenuator" shown in Figure 3.22. As discussed in Section 4.2, a "variable rf phase shifter" removes the phase variation of this variable rf attenuator, so that just the amplitude of the rf signal driving the shaping AOMs can be varied.

a frequency f_{sweepup} to 20 MHz on the rising edge of its external trigger which is connected to Y2. Similarly, the chirp down synthesizer sweeps between $f_{\text{sweepdown}}$ and 20 MHz when it receives a rising edge from Y1. Depending on the beam direction, the output of one of these synthesizers is used to generate the 40 MHz for both shaping AOMs. A copy of the logic output X1 and its inverse $\bar{X1}$ goes to a *MiniCircuits* model ZYSW-2-50DR rf switch which selects between the outputs of these two synthesizers. If X1 is active, the beam direction is inverted and the chirp down synthesizer is used. On the other hand, if X1 is inactive, indicating that the switchyard is set to normal (or off), the chirp up synthesizer is used.

3.6.4 Timing

To guarantee that the timing of the interferometer patterns repeats exactly the same each time, we must insure that 1) all of the synthesizers which generate the Raman beams use oscillators traceable to the same timing reference and 2) the trigger signal which starts the pattern is synchronized with these oscillators. The *Stanford Research Systems* model DS345 and the *Hewlett Packard* model HP33120A synthesizers both have internal 40 MHz clocks locked to an external 10 MHz reference traceable to LORAN C. The DDS is based on a 1 GHz clock also based on LORAN C. The only other oscillator is the internal clock of the AWFG board, which normally comes from an internal 20 MHz oscillator. We disabled this internal oscillator and replace it with an external signal derived from the LORAN C 10 MHz reference.

To satisfy the second condition, in principle we could synchronize the incoming trigger signal with any of the oscillators or even with the 10 MHz reference signal itself. Consider two oscillators, one at frequency f and the other at frequency $4f$, and an incoming trigger edge that drifts with respect to the oscillators. If we force the trigger edge to occur at one of the rising edges of the $4f$ signal, it may not repeat with respect to the clock at frequency f. Relative to this slower clock, there are four different possible phases at which the trigger could occur. However, if we synchronize the trigger with the slower clock, the trigger edge will always repeat with respect to the faster clock. Therefore, we always synchronize the trigger with the slowest oscillator in the experiment, which in our case is the divided down AWFG board's sample clock. On the AWFG board, the 20 MHz clock is divided by four in two stages to derive a master sample clock oscillating at 5 MHz. Depending on the desired sample rate for the stored waveform, this clock is then divided down by up to two 16-bit integers. Most of our data were taken with the AWFG sampling at 1 MHz (master clock divided by 5). This 1 MHz clock is the slowest clock in the experiment. To synchronize the incoming trigger signal with this clock we use a TTL latch. If we clock the latch with a copy of the 1 MHz AWFG sample clock and put the trigger signal into the data input, the output will change state to reflect the input only on one of the 1 MHz clock rising edges. If we use this output to trigger the Raman pattern, we know it will repeat in exactly the same way with respect to all of the oscillators used to generate the pattern.

We now present the timing diagrams for generating three representative cases of real interferometer sequences. We first show the complete timing diagram for all four interferometer geometries with $T = 5$ ms and 30 π-pulses. For this case, Figure 3.25 shows a representation of the atomic trajectory overlayed on the master timing diagram, identifying the regions when (a) the atoms are loaded into the MOT, (b) they are launched, (c) the interferometers are built, and finally (d) the atom signal is detected. Figures 3.26 through 3.29 zoom in on the interferometer sequence shown at the bottom of Figure 3.25 for geometries $\boxed{1}$ through $\boxed{4}$. Typically, for an interferometer sequence $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{1}\ldots$, the master timing diagram (Figure 3.25) would repeat with each consecutive repetition generating the next interferometer geometry: Figure 3.26 then 3.27, 3.28, 3.29, 3.26, and so on. Finally, Figures 3.30 and 3.31 show the master timing diagrams for long interferometers with $T = 120$ ms with 30 and 0 π-pulses, respectively.

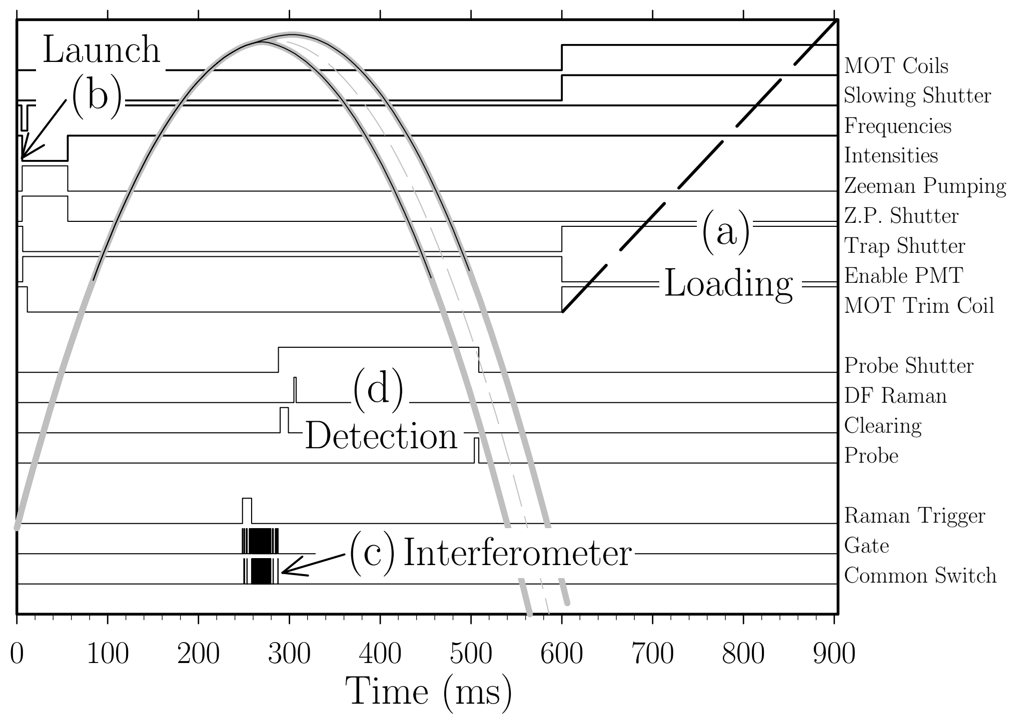


Figure 3.25: **Example 1: Timing diagram for interferometers with 30 π-pulses and $T = 5$ ms.** The story starts when the cesium atoms are loaded (a) into a magneto-optic trap (MOT). At $t = 5$ ms, the atoms are launched (b) vertically. The parabolic curve (shown in wide gray) represents the atoms vertical position. Due to the momentum recoils from the π-pulses an interferometer and its conjugate are pushed down and up, respectively, away from the gray dashed line which shows the atomic trajectory with no recoils. When the atoms enter the magnetic shielding (represented on the trajectory with a narrow dark line), the interferometer sequence is triggered. This "Raman trigger" is shown at the bottom with the "gate" and "common switch" controls, from which one can identify where each of the (in this case) 35 adiabatic transfer pulses occur. (The following four figures zoom in on this region for each of the four different interferometer geometries). A particular interferometer geometry is constructed (c). Finally, after the atomic cloud leaves the magnetic shielding, the number of atoms emerging from the interference is detected (d). Every 908 ms when this sequence repeats with the same loading and launch sequences, the remaining interferometer controls (not shown) are reprogrammed and the next interferometer geometry is built.

Figure 3.26: **Timing diagram generating geometry 1 with 30 π-pulses and $T = 5$ ms.**

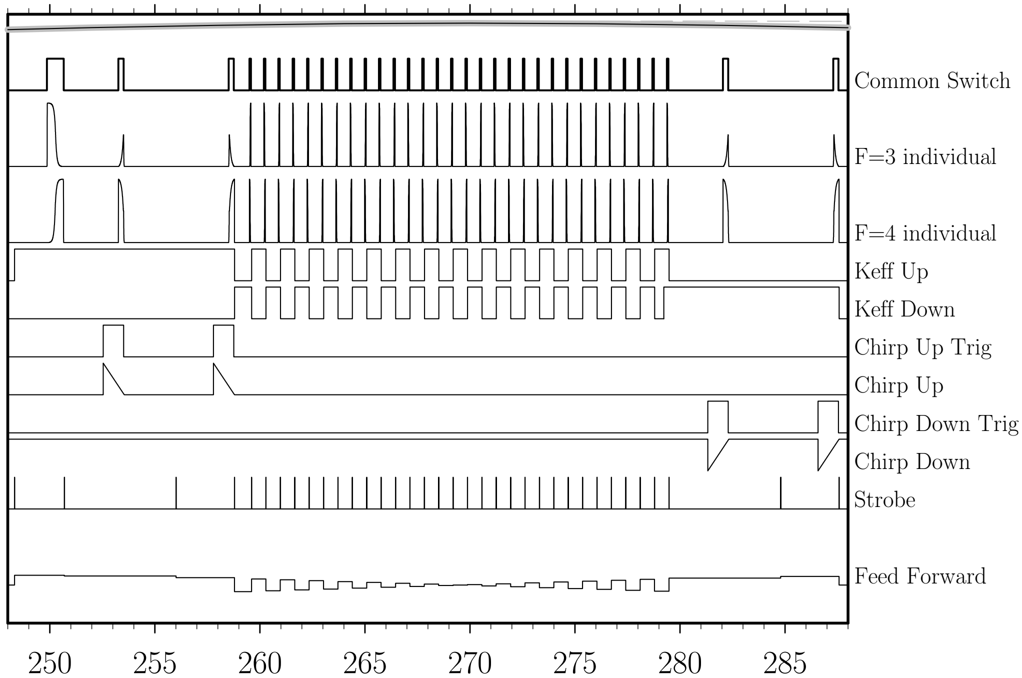


Figure 3.27: **Timing diagram generating geometry 2 with 30 π-pulses and $T = 5$ ms.**

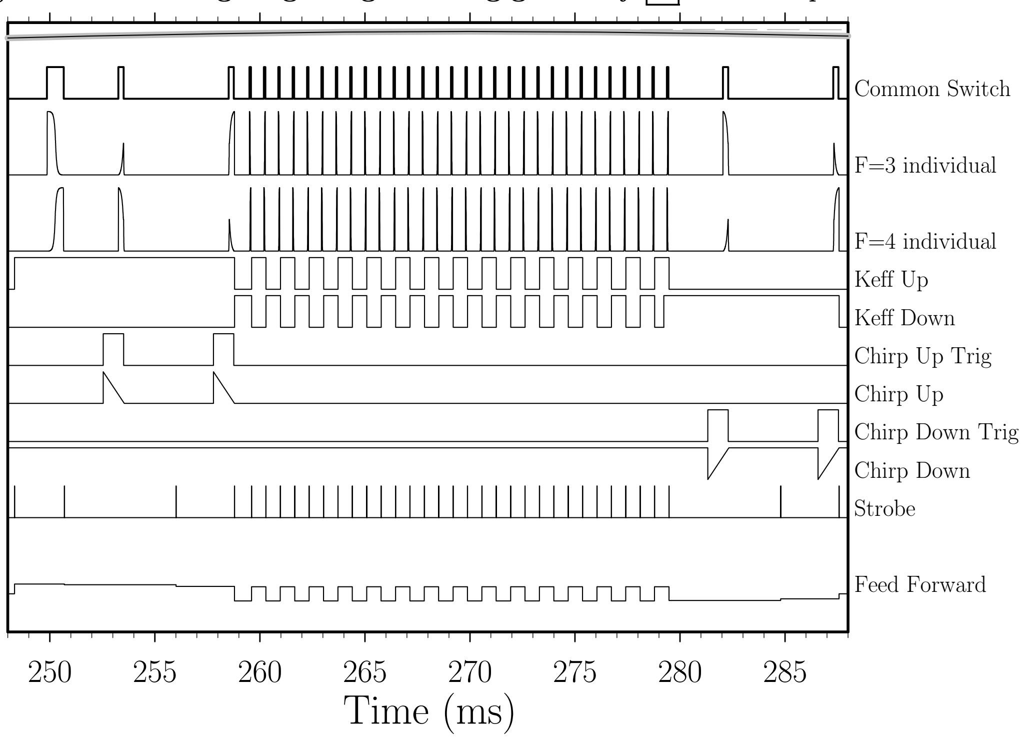


Figure 3.28: **Timing diagram generating geometry** $\boxed{\mathbf{3}}$ with 30 π-pulses and $T = 5$ ms.

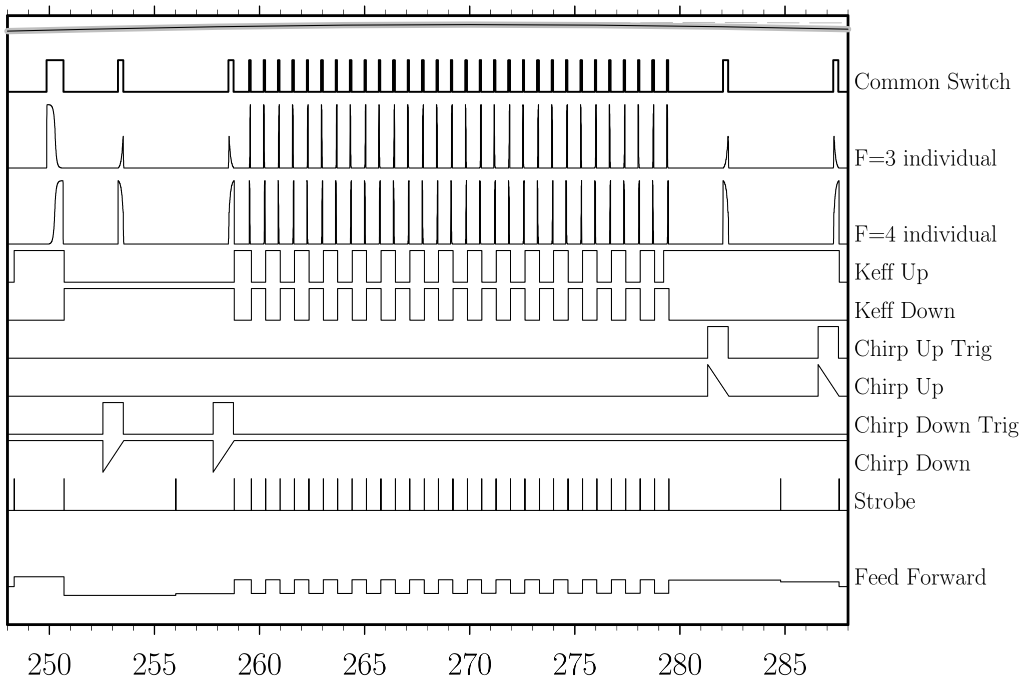


Figure 3.29: **Timing diagram generating geometry** $\boxed{\mathbf{4}}$ with 30 π-pulses and $T = 5$ ms.

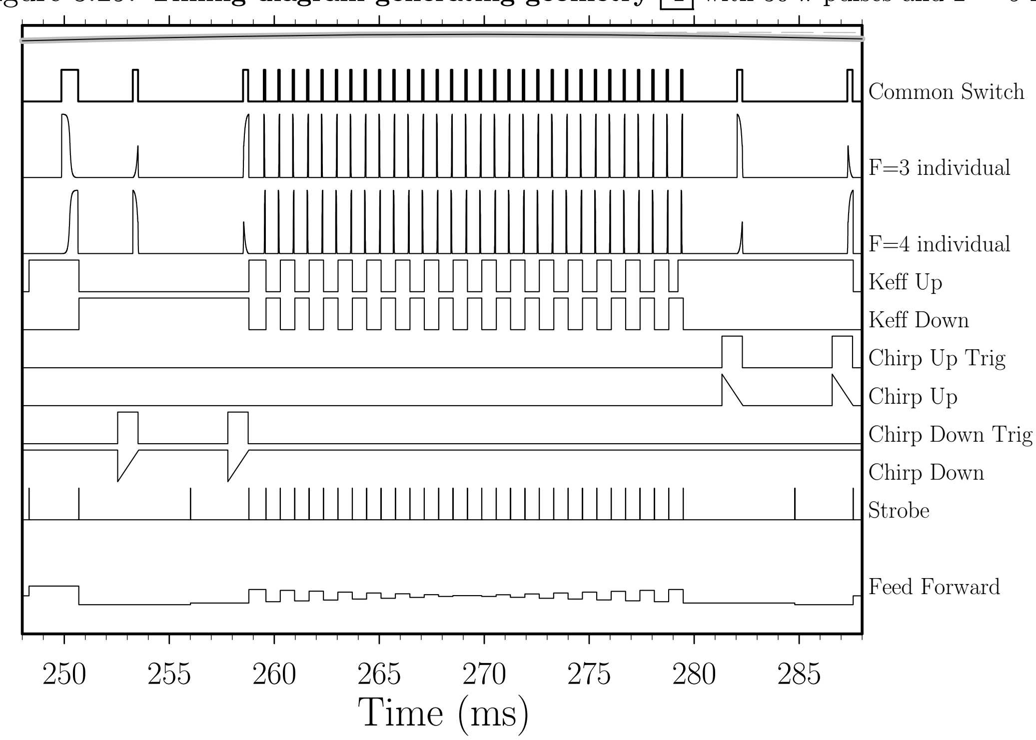


Figure 3.30: **Example 2: Timing diagram for interferometers with 30 π-pulses and $T = 120$ ms.**

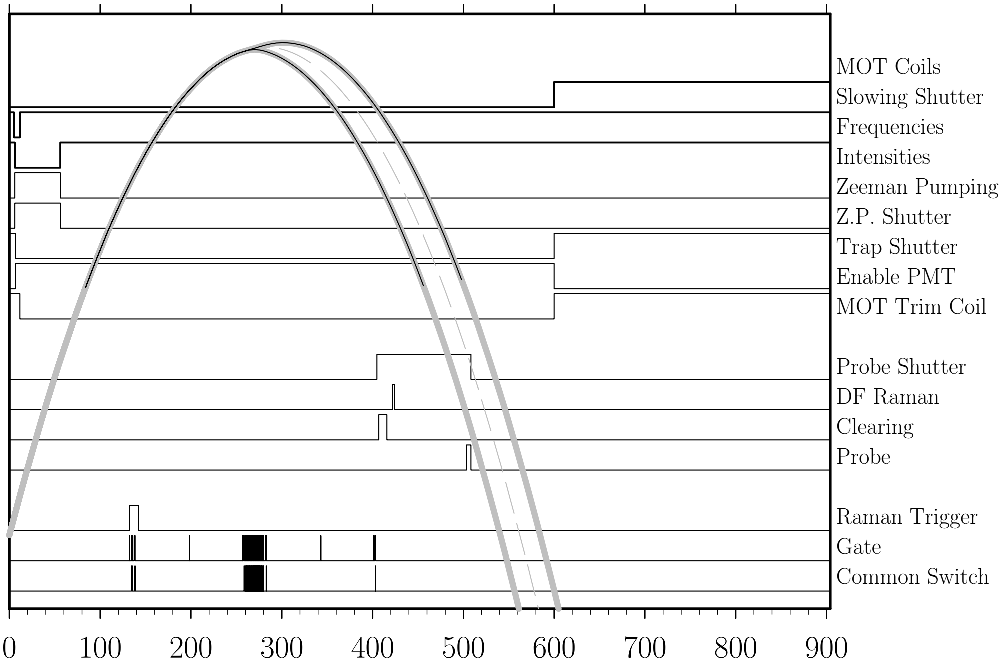
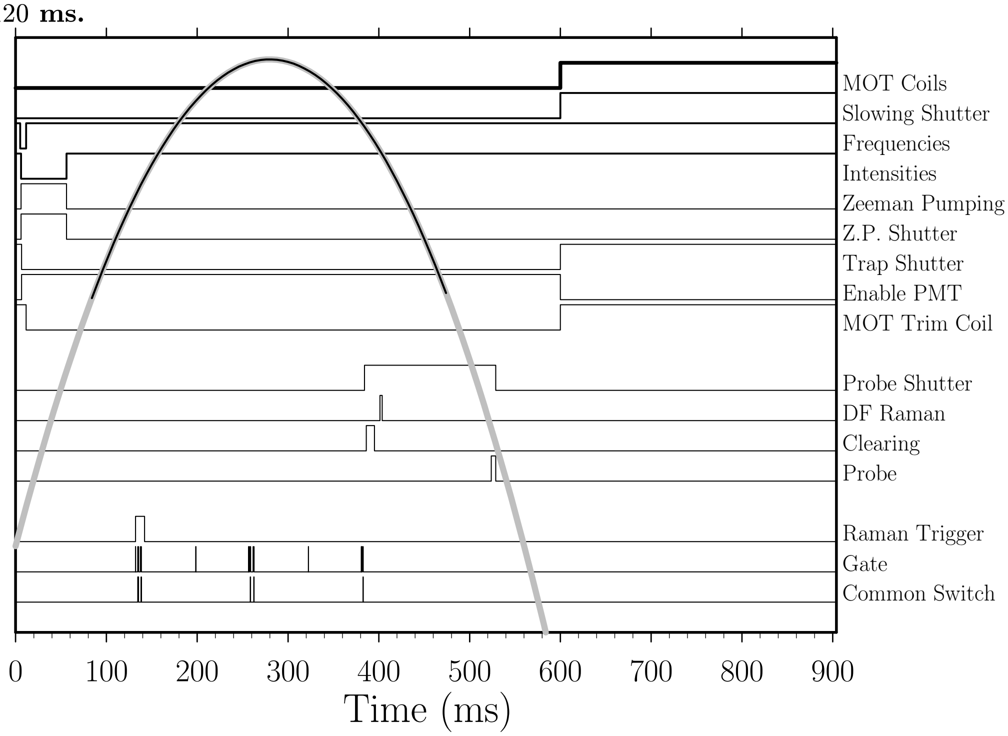


Figure 3.31: **Example 3: Timing diagram for interferometers with no π-pulses and $T = 120$ ms.**



3.7 Tests of adiabatic passage

3.7.1 π-pulses

To demonstrate adiabatic transfer, we launch the atoms from the MOT as normal. They leave the MOT in the $F=4$ state, arbitrarily distributed among the $2F+1=9$ magnetic sublevels. If the Zeeman pumping is activated, the $F=4, m_F=0$ signal is increased by roughly a factor of 3 through optical pumping, leaving the remaining atoms in the other $F=4, m_F \neq 0$ sublevels. We program the Raman beam synthesizers to generate a single adiabatic π-pulse with the shape shown in Figure 2.7. At some point along the trajectory when the atoms are inside the magnetic shielding, this single pulse sequence is triggered. The adiabatic π-pulse transfers a fraction of the atoms to the $F=3$ state, leaving the rest of the atoms in the $F=4$. After this pulse, the clearing laser turns on. Atoms in the $F=4$ state will absorb a single photon and recoil upward with the clearing beam direction. Once in an excited state they spontaneously emit a photon and recoil in a random direction. From the $F=5'$ excited state, they can fall back only into the $F=4$ ground state where they start the process all over again. Since the clearing laser is tuned to the closed $F=4 \rightarrow F=5'$ D2 transition, this process continues until the atoms have either 1) moved out of the beam due to the scattering of random photons or 2) fallen into the $F=3$ ground state due to off-resonant ($\Delta \sim 250\,\mathrm{MHz} = 5\Gamma$) coupling to the $F=4'$ excited state. In either case and also because the atoms are pushed upward by the clearing beam, none of these atoms will be detected by the probe laser which is tuned to the same atomic transition. After the clearing pulse, there are no atoms to be detected in the $F=4$ state. The signal atoms are in the $F=3, m_F=0$ state along with some remaining unwanted background atoms in the other $F=3$ magnetic sublevels.

The next light pulse comes from the DF Raman laser, which transfers a significant fraction of the $F=3, m_F=0$ atoms to the $F=4, m_F=0$ state, leaving the atoms in the $F=3$ $m_F \neq 0$ states untouched. Since the two frequencies comprising the DF Raman beam copropagate, the atoms do not recoil and continue along their original trajectory to be detected by the probe pulse. Like the clearing laser, the probe is tuned to the closed $F=4 \rightarrow F=5'$ transition, so it causes any atoms in any $F=4$ magnetic sublevel to scatter photons over and over. A fraction of these scattered photons are collected by a lens and focused into a photomultiplier tube (PMT), which finally converts them into a signal that we can observe and record. The size of this signal is proportional to the number of atoms in the $F=4$ state. However, because of the magnetic state selective transfer, the only atoms in the $F=4$ state at the time of the signal recording originally came from the signal atoms in the $F=3, m_F=0$ state. None of the background atoms in the other $F=3$ magnetic sublevels are detected.

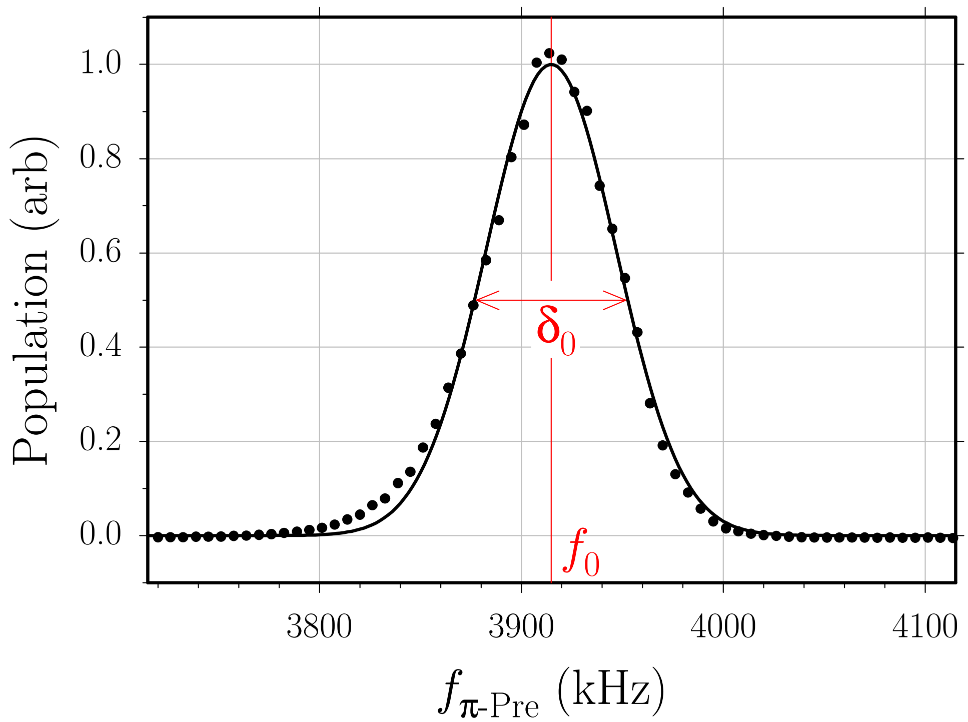


Figure 3.32: **Adiabatic transfer using a single velocity preselecting π-pulse.** The π-pulse is 800 μs long and reduced to 25% of the maximum possible light intensity. For each launch the difference frequency between the two Raman lasers is changed and the number of atoms emerging from the π-pulse is detected. The data are fit by the function $\exp[-4\ln(2)(f-f_0)^2/\delta_0^2]$, giving $f_0 = 3914.72(29)$ kHz and $\delta_0 = 76.021(75)$ kHz. Since the linewidth of the transfer pulse (FWHM$\sim$ 37 kHz) is less than the observed width of the resonance, δ_0 is predominantly determined by the width of the atomic velocity distribution.

We now repeat this entire process, but for each subsequent launch we change the Raman laser difference frequency δ. The adiabatic π-pulse is 800 μs long and reduced to 25% of the maximum possible light intensity, and thus has a linewidth of 37 kHz (see below). The measured rms velocity of the atoms launched from the MOT is $v_{\mathrm{MOT}} = 1.4$ cm/s (see Section 3.1.4). This velocity is equivalent to a two photon detuning of $v_{\mathrm{MOT}}/\lambda_{\mathrm{eff}} = 31$ kHz and a linewidth 74 kHz. Therefore, because the linewidth of the adiabatic π-pulse is narrower than the Doppler width of the atoms, the width of the lineshape shown in Figure 3.32 is predominantly determined by the width of the atomic velocity distribution. By fitting a Gaussian lineshape to this distribution, we can find the lineshape center to within ~ 0.3 kHz. We thus determine the mean atomic velocity along the Raman beams at the time of the pulse to within 0.1 mm/s. Furthermore, because all subsequent adiabatic transfer pulses have a broader spectral width, this first π-pulse selects the particular velocity class from the atomic velocity distribution that will participate in the interferometers.

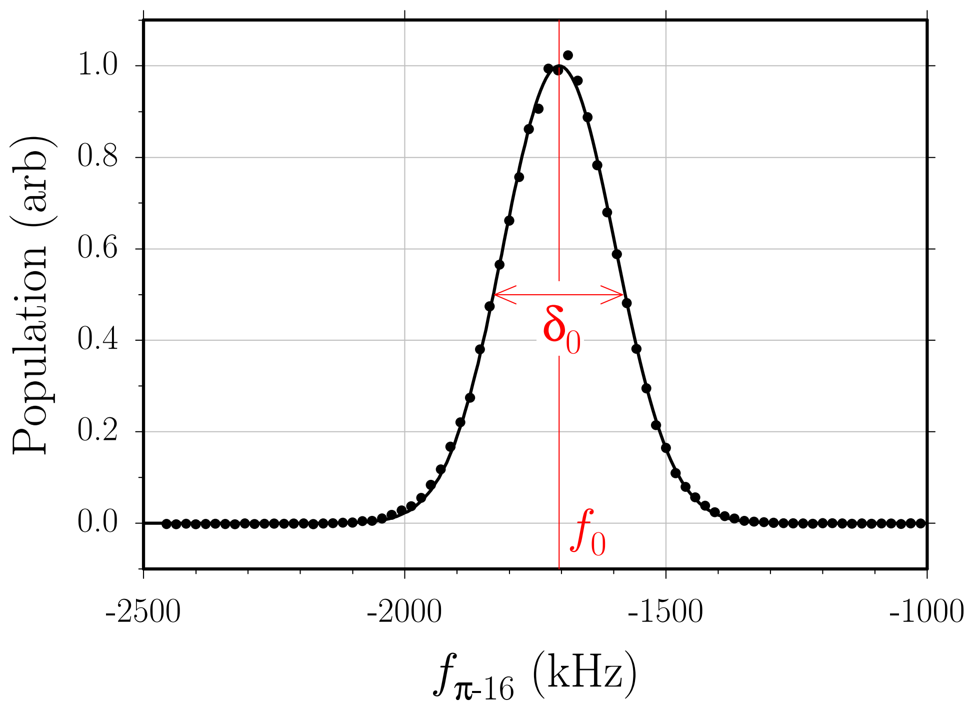


Figure 3.33: **Adiabatic transfer linewidth.** Sixteen 70 μs long π-pulses follow a single velocity preselection π-pulse. For each launch the difference frequency between the two Raman lasers for only the last π-pulse is scanned and the final number of atoms detected. The data are fit to the function $\exp[-4\ln(2)(f - f_0)^2/\delta_0^2]$, giving $f_0 = -1704.29(38)$ kHz and a FWHM $\delta_0 = 252.67(98)$ kHz. Since the atomic velocity distribution was narrowed by the preselection π-pulse, the observed width of this resonance represents the linewidth of a 70 μs long π-pulse.

To verify the linewidth of the adiabatic transfer pulses, after this long low intensity velocity "preselection pulse", we program sixteen normal π-pulses at full intensity lasting only 70 μs . By scanning the laser difference frequency of the last of these regular π-pulses, we can map out the pulse linewidth shown in Figure 3.33. This lineshape can be described by the Gaussian

$$g(f) = A \exp\left(-4\ln(2)\frac{(f-f_0)^2}{\delta_0^2}\right) + B \tag{3.22}$$

where A and B are arbitrary amplitude and offset factors and δ_0 is the FWHM linewidth. Fitting this lineshape to the data in Figure 3.33 results in a linewidth $\delta_0 = 253$ kHz. By varying the length τ and intensity I of these pulses, we verify that the linewidth δ_0 of adiabatic transfer pulses scales according to

$$\delta_0 \propto \sqrt{\frac{I}{\tau}} \tag{3.23}$$

Thus, because the preselection π-pulse is 4 times less intense and $800/70 = 11$ times longer, its linewidth is $(253\,\text{kHz})\sqrt{0.25/11} = 37$ kHz. Without this velocity preselection the interferometer contrast is worse, because there is a larger background level of atoms which do not participate in the interferometer because their velocity Doppler shifts them too far from resonance for the $\pi/2$-pulses to address them.

By observing how the signal shrinks as we add more regular π-pulses after the single velocity preselection π-pulse, we determine an average transfer efficiency of $\epsilon = 93.8\%$. After N π-pulses with efficiency ϵ, only a fraction ϵ^N remain. Even with pulses as efficient as ours, after 30 π-pulses, only $(0.938)^{30} = 0.147$ of the atoms remain, and a drop in efficiency of only 2% reduces this number almost by half. The dominant limit on the transfer efficiency of the π-pulses is the selective detection of only the $m_F\!=\!0$ atoms. Even with a magnetic bias field of ~ 72 mG that shifts the $m_F\!=\!1$ levels apart by 50 kHz, the π-pulses still transfer atoms between the $F\!=\!3, m_F\!=\!+1$ and $F\!=\!4, m_F\!=\!+1$ states. Since only the atoms in the $m_F\!=\!0$ states are detected, the transfer efficiency is artificially lower. With the magnetic bias field set to zero, the π-pulse efficiency is $\sim 96\%$, but these π-pulses are not useful for interferometer measurements that require avoiding the magnetic field sensitive sublevels.

3.7.2 Interferometry

To build interferometers with adiabatic transfer pulses, we add four $\pi/2$-pulses to the pulse sequence. After the preselection π-pulse which precedes all pulse sequences, we wait $\sim$3 ms to give the tracer phaselock plenty of time to settle before we start the first interferometer $\pi/2$-pulse. The first $\pi/2$-pulse transfers the atoms from the pure $F=3$ state to a superposition of the $F=3$ and $F=4$ states. From the end of the first $\pi/2$-pulse, this atomic state evolves freely for a time T before the second $\pi/2$-pulse starts. Because the two interferometer paths do not overlap at the second $\pi/2$-pulse, this pulse projects half of the atoms onto the bright state, which scatter incoherent photons and leave the interferometer[9]. The remaining half of the atoms projected into the dark state emerge from the second $\pi/2$-pulse in the $F=3$ state for interferometer geometries $\boxed{1}$ and $\boxed{3}$ and in the $F=4$ state for interferometers $\boxed{2}$ and $\boxed{4}$. Both interferometer paths then experience N regular π-pulses. Before each π-pulse we wait a time $T_{\pi\pi}$. After time t gravity changes the atomic velocities by $\Delta v = -gt$. We choose $T_{\pi\pi}$, the delay between π-pulses, so that this velocity change due to gravity is equal to one two-photon recoil v_r: $T_{\pi\pi} = v_\mathrm{r}/g = (6.71\,\mathrm{mm/s})/(9.80\,\mathrm{m/s}^2) = 685\,\mu\mathrm{s}$. The net effect of the upward pushing π-pulses is thus to cancel the acceleration of gravity, while the downward pushing π-pulses accelerate the atoms downward at $2g$. After the last π-pulse, before starting the third $\pi/2$-pulse, we again wait $\sim$3 ms for the tracer phaselock loop to settle. After the third $\pi/2$-pulse pulse, each interferometer path is once again in a superposition of the two pure hyperfine ground states. The atoms again evolve freely for time T before the fourth $\pi/2$-pulse starts. At this point, the overlapping paths interfere and emerge from the fourth $\pi/2$-pulse in the $F=3$ state with an amplitude that depends on the total phase difference accumulated between the two interferometer paths. The non-interfering paths also emerge in the

[9] It is not *strictly* true that they leave the interferometer. Many of the atoms absorb a photon, fall back into the signal dark state, and then continue through the remaining light pulses to be detected at in the end. On average, however, these atoms contribute no net phase to the final fringe signal, although as discussed in Section 2.3.1, they do reduce the contrast.

$F = 3$ state with some amplitude. The interferometer is now complete. Since the interference has already occured, the atoms are no longer sensitive to phase shifts. The only remaining task is to count the atoms and record the signal. As described above and in Section 3.7.1, in order to detect only the atoms in the $F = 3, m_F = 0$ state, we first blast all of the $F = 4$ atoms away. A single DF Raman π-pulse then transfers the $F = 3, m_F = 0$ atoms to the $F = 4, m_F = 0$ state, which can be detected by the probe beam.

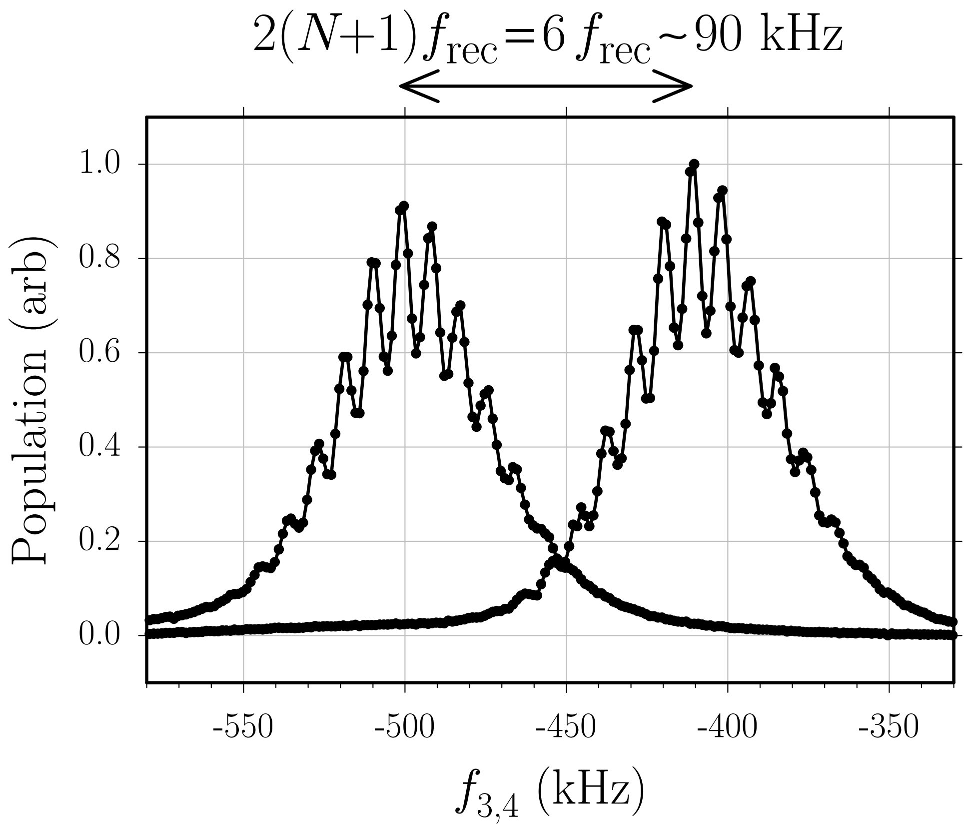


Figure 3.34: **Fringe structure from four $\pi/2$-pulse interferometers.** The first two and last two $\pi/2$-pulses are separated by the free evolution time $T = 0.1$ ms, which determines the period of the fringes $1/T = 10$ kHz. The two interferometers, which each produce one of the two resonances, are further separated by $N = 2$ additional π-pulses. The frequency separation between these two resonances is a known integer multiple (in this case, 6) of the recoil frequency f_{rec} we are trying to measure. The π-pulses linearly amplify the size of the separation while the $\pi/2$-pulses superimpose fringes on top of the resonances. These fringes allow us to much more precisely determine the centers and thus the separation of the two resonances. In order to make a high sensitivity measurement of f_{rec}, T can be increased to as long as 160 ms and as many as 50 π-pulses can be added to each interferometer, causing a separation of over 200 optical photon recoils.

Similar to the graph represented in Figure 1.3, the $\pi/2$-pulses serve to superimpose phase sensitive fringes with periodicity $1/T$ onto a resonance. In order to see how the fringes appear on top of the resonance lineshape, Figure 3.34 presents data from a relatively insensitive interferometer condition when $T = 0.1$ ms and $N = 2$. As described in Section 1, the goal of the measurement is to measure the difference between the center of two resonances that are separated by a integer number of recoil frequency shifts. With 30 π-pulses the resonances are separated by 62 two-photon recoils or 124 single optical photon recoils, equivalent to a shift of almost 1 MHz. Thus, to make a part per billion measurement with 30 π-pulses, we must find the center of the two roughly 100 kHz wide resonances to within ~ 1 mHz. In other words, without the fringes, we would have to split the resonance lines by one part in 10^8. With the fringes, however, we have a much finer feature we can use to find the resonance center. Since T can be made as long as $\sim$160 ms before the interferometers no longer fit inside the magnetic shielding, the fringe spacing can be made as small as $1/(2T) \sim 3$ Hz, which in one step improves our accuracy by over four orders of magnitude.

Chapter 4

Improvements

As part of testing and thoroughly evaluating the previous experimental setup [9], we have identified several problems and implemented solutions. We determined that the largest systematic error (over 100 ppb in f_{rec}) was due to uncharacterized systematic phase shifts from the radio-frequency (rf) synthesizer. In Section 4.1 we discuss this problem and the improved performance of the replacement synthesizer. In addition to the phase errors from the synthesizer, it was also clear that our method of switching the Raman light on and off introduced a systematic and repeatable phase shift, predominantly due to the intrinsic phase variation as the attenuation of the variable rf attenuators changed. In Section 4.2 we discuss the feed forward system we designed and built to reduce this phase versus rf amplitude variation by over an order of magnitude. We further reduced the phase errors from intensity switching by installing an additional acousto-optic modulator (AOM). This "common switch" AOM switches both Raman beams on or off together in exactly the same manner and at precisely the same time. Adding this AOM removed a systematic error of over 40 ppb from the $\pi/2$-pulses discussed in Section 6.7. Related to this problem of phase errors during intensity switching, we also observed repeatable fluctuations in the light intensity diffracted by the shaping and common switch AOMs. In Section 4.7 we examine these fluctuations and the improved response after replacing the rf amplifiers with ones better suited for switching.

By improving the efficiency of the Raman beam generation, we were able to increase the available amount of Raman beam power by a factor of approximately three. We also improved the spectral purity of many of our frequency references by installing crystal filters (Section 4.6).

We improved our alignment procedures to have better control over the Raman beams: their collimation (Section 4.3), their relative angular alignment (Section 4.4), and the matching of their intensities (Section 4.5).

In an effort to improve the signal to noise ratio of the final interferometer signal[1], we investigated the phase noise added by the two optical fibers used to spatially filter the Raman beams (see Section 3.2.6). Because the tracer laser also passes through these fibers, the tracer phase lock loop removes most of this noise up to the limit where the phase noise added to the Raman beams is not the same as the noise applied to the tracer. Our results indicated that the remnant differential phase noise added by the fibers might represent a significant amount of the total noise on the final atom signal. As an attempt to reduce this noise, we changed the wavelength of the tracer laser from 887 nm to 896.6 nm, thus reducing the detuning from the cesium resonance from -8 nm to $+2$ nm and presumably improving the common mode reduction of the fiber phase noise.

We also believe that a significant fraction of the phase noise of the interferometer signal comes from the motion of the top polarizing beamsplitter (PBS) cube not removed by the vibration isolation (VI) system discussed in Section 3.4. As an attempt to improve the performance of this system, we redesigned and rebuilt the mechanical support structure. In particular, we switched the position of the air piston and the acceleration sensor so that the sensor is now closer to the interferometer platform (Figure 3.14) which we are trying to isolate from external vibrations. We were particularly concerned about the rigidity of the air piston. Because of the finite air pressure, the inner cylinder of the piston will always be able to tilt slightly away from the axis of the outer sleave. If the accelerometer sits above the air piston and the entire VI column tilts about the piston, the motion detected by the sensor will be opposite in sign from the actual motion of the interferometer platform. Because

[1] The random noise of the final atom signal is discussed further in Section 5.2.

the sign is opposite, as the feedback loop attempts to reduce the motion of the sensor, it will actually increase the motion of the interferometer platform. With the accelerometer on the same side of the air piston as the interferometer platform, the action of the feedback will always be to reduce the effects of rotational motion on the interferometer platform.

As an additional precaution we installed a model 755-1129 tilt sensor from *Applied Geomechanics* on the freely swinging VI column. By rotating the column about the axis of the air piston, we calibrated the sensor to find its total offset from vertical. By tilting the outer sleave of the air piston until the output of the tilt sensor matched this level point, we were able to make the axis of the air bearing vertical to better than 100 μrad. By temporarily installing a tilt stage under the accelerometer, we verified that the sensitive axis of the accelerometer is also vertical. Since the Raman beams whose wavefronts we are trying to stabilize are also aligned to be vertical, we thus insure that the isolation axis of the VI system is parallel to the Raman beams.

4.1 RF synthesizer

In the scheme shown in Figure 3.12, the optical phase of the interferometer laser fields is locked to a radio-frequency (rf) oscillator. By changing the frequency and phase of this reference oscillator before every light pulse, we control the frequency and phase of the interferometer laser fields, which act as the reference to be compared with the internal phase of the atoms (see Section 2.1). The frequency is set to maintain the two-photon resonance condition with the atoms. This two-photon resonance frequency changes by as much as approximately ± 4 MHz over the whole usable atomic trajectory due to gravity and by f_{rec} per π-pulse, or $\sim$600 kHz for 40 π-pulses. Because the π-pulses might be separated by only a few hundred microseconds, the oscillator should have a switching time no longer than $\sim 10\,\mu$s. Furthermore, because the oscillator serves as the reference in an interferometry experiment, its exact phase must be known for all times. In other words, in a predictable way it must switch phase continuously between two frequencies defined exactly in terms of some repeatable time standard.

A direct digital synthesizer (DDS) is ideal for this application. Figure 4.1 shows

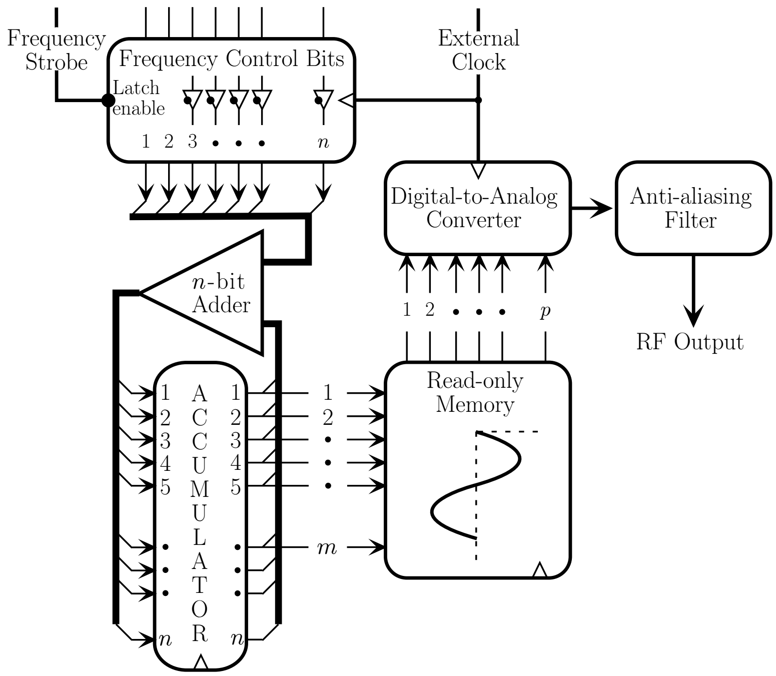


Figure 4.1: A **direct digital synthesizer (DDS)** outputs a sinusoidal signal whose frequency is an exact integer submultiple of the frequency of an external clock signal. The integer that determines the output frequency is set with n binary "frequency control bits". The value of this integer controls the rate at which an adder plus an accumulator step through the 2^{m} addresses of an internal read-only memory into which is stored a digitized version of a single period of a sine-wave. A digital-to-analog converter converts the p-bit output of the memory into an analog signal. Because their output is both frequency tunable and phase locked to the external clock signal, DDSs are ideal for use in precision interferometry.

a diagram of the fundamental components of a DDS. An external clock signal of frequency f_{clk} drives an n-bit binary counter called the accumulator. The m most significant bits of the accumulator address a digital read-only memory, which contains a p-bit digitized representation of a single period of a sine-wave. The binary values loaded from the memory are converted to an analog signal level by a fast DA-converter followed by an anti-alias filter. Given a binary number P from 0 to $2^p - 1$, the memory plus the DA-converter converts this number into an output voltage level of $A \sin(2\pi P/\ 2^p)$, where A is some arbitrary voltage amplitude. The anti-alias filter smooths the output sine wave and thus reduces the leakage of the clock frequency into the final output. At every clock cycle an n-bit number N is added to the accumulator. The number N is determined by the state of the n digital external frequency control inputs. These n digital frequency control lines thus set the output frequency of the synthesizer to an exact output sub-multiple of the clock frequency. For example, if the accumulator starts at zero and $N = 2^{(n-2)} = (N_{\max}+1)/4$, at the first active clock edge, the accumulator will change from 0 to $2^{(n-2)}$. The top m bits will change by $2^{(m-2)}$ or $(M_{\max}+1)/4$ or one quarter of one cycle. At the next active clock edge the accumulator will advance another quarter cycle through the sine-wave. The output frequency will thus be exactly $f_{\text{clk}}/4$. For arbitrary N, the output frequency will be Nf_0, where $f_0 = f_{\text{clk}}/2^n$ is the smallest frequency step.

The DDS unit we use, model ADS-431-1367 from *Sciteq Communications*, has 32 digital frequency control lines ($n = 32$) and can accept clock signals as fast as 1.6 GHz. Its internal anti-alias filter is 6th order with a 3 dB corner frequency of $\sim$ 500 MHz. Since m and o are internal to the device and only limit the amplitude purity, they are specified as a spurious peak limit of -45 dBc "typical" over its entire output range. We operate this unit with a clock frequency of 1.0 GHz and an output frequency centered at 12.631 770 MHz. At these frequencies, we observe a typical spurious peak level of less than -60 dBc.

For our purposes, however, the *phase* purity of the DDS output is by far the most critical performance specification. A particularly attractive feature of DDS units is that their output phase noise is as good as the input clock signal, even when switching between two output frequencies. Neglecting the small amount of amplitude noise added by the device, the phase noise at a single output frequency f_{out} is given by the phase noise of the clock signal divided by the ratio $f_{\text{clk}}/f_{\text{out}}$. Also, during an output frequency change, because the time derivative of the accumulator changes and not the accumulator itself, the output always changes phase continuously. Unfortunately, because this synthesizer must change output frequencies during an interferometer, besides random phase noise we are also concerned about any repeatable deviation from a instantaneous frequency change. Specifically, if the output frequency of the DDS is to change from f_1 to f_2 at time t_{12}, what will the phase ϕ_2 be at some later time $t_2 > t_{12}$ compared to an initial phase ϕ_1 defined at some earlier time $t_1 < t_{12}$?

Ideally, the answer is

$$\Delta\phi = \phi_2 - \phi_1 = 2\pi\left[f_1(t_{12} - t_1) + f_2(t_2 - t_{12})\right] \tag{4.1}$$

In practice, however, there will always be some delay between the signal requesting the frequency change at t_{12} and the time when the synthesizer actually makes the change. If this delay is given by $\tau + \alpha f$, the above expression for the phase difference must be modified to

$$\Delta\phi = 2\pi\left[f_1(t_{12} - t_1) + f_2(t_2 - t_{12})\right] - 2\pi\left[\tau(f_2 - f_1) + \frac{1}{2}\alpha(f_2 - f_1)(f_2 + f_1)\right] \tag{4.2}$$

In this expression, α represents a non-linear delay that depends on the synthesizer's output frequency, while τ is the zero-order delay that might, for instance, come from the propagation delay through a simple transmission line. Any dispersive element located between the DA-converter inside the DDS and the phase detector in our experiment, whose transmission delay depends on the frequency, could result in a non-zero α. By far the largest source of such non-linear phase delays comes from any filter on the output of the DDS. For this reason, we intentionally do no filtering of the DDS output, thereby insuring that its dynamic phase performance is limited only by its internal anti-aliasing filter.

To fully characterize the performance of this DDS unit, we recorded its output with a fast digital, sampling oscilloscope around a frequency transition from f_1 to f_2. By fitting the data before and well after the frequency change with sinusoidal functions, we were able to calculate a phase change $\Delta\phi = \phi_2 - \phi_1$. Figure 4.2 shows these measurements with $f_1 = 250$ MHz and different values of f_2 from 10 to 250 MHz. Note that the results of this direct measurement (open circles) exhibit discrete jumps as a function of f_2 that cannot be explained by the model in equation (4.2). Upon further investigation we found qualitatively the same jumps over all frequency ranges, always occuring at frequencies which are exact binary sub-multiples of the clock frequency (125 MHz, 62.5, 31.25, *etc.*). Based on a suggestion from the manufacturer, we suspected that the individual frequency control bits have different propagation delays through the device before they affect the output. In particular,

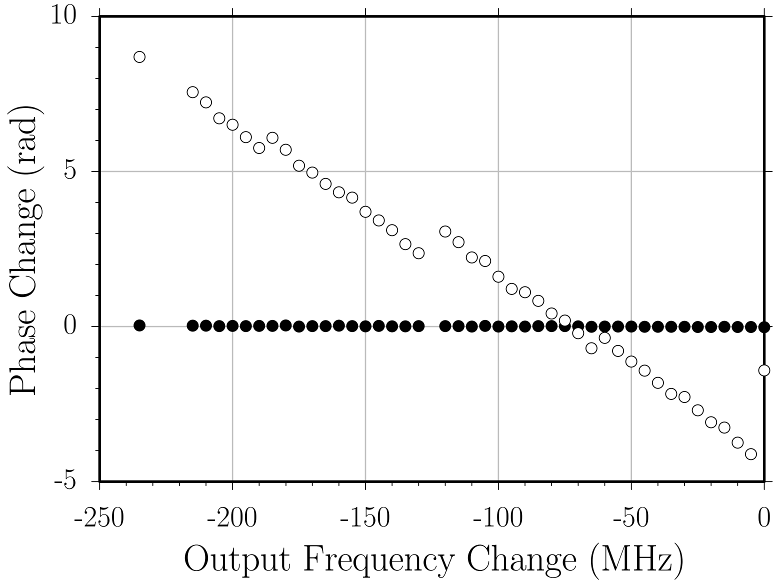


Figure 4.2: **Systematic phase error from the direct digital synthesizer (DDS)** when it switches output frequency from f_1 to f_2. In this case, $f_1 = 0$ and f_2 varies from 250 to almost 10 MHz. The synthesizer's output is sampled before and well after the frequency change and fit with two sinusoidal functions whose phase difference is plotted on the vertical axis. Over $\Delta f = f_2 - f_1 \simeq 250$ MHz, this phase error (hollow circles) can be well over one cycle. However, this error is predictable and can be subtracted out (solid circles) using a simple rule (equation 4.9) for the propagation delay of the individual frequency control bits. A fixed propagation delay of 42 ns is subtracted out of both data sets. Such a delay independent of Δf appears as a slope in a graph like this of phase versus Δf and does not affect the recoil experiment.

we proposed that the additional time it takes before a change at bit b affects the output frequency is given by $(32 - b)/f_{\mathrm{clk}}$. Thus, the least significant bit will take 31 clock cycles longer to affect the output than the most significant bit. Although difficult to understand from a design standpoint, correcting for this behavior has proved relatively simple. To calculate the net phase shift due to this effect, we first assume that the DDS is switching between frequency $f_1 = 0$ and f_2. Since $f_1 = 0$, all frequency control bits must be off to begin with. If this variable delay model holds, then j clock cycles after the 32 frequency control bits for f_2 are applied, the output frequency $f(t_j)$ and phase $\phi(t_j)$ of the device will be

$$f(t_j) \quad = \quad f(t_{j-1}) + \sum_{b=1}^{32} \ \delta_b(f_2)\ 2^{b-1}\ f_0\ \delta_{j,33-b} \tag{4.3}$$

$$\phi(t_j) \;=\; \phi(t_{j-1}) + \frac{2\pi}{f_{\rm clk}} f(t_j) \qquad (4.4)$$

where $\delta_b(f) = 1$ if bit b in the frequency control bits for output frequency f is on, $\delta_{j,33-b} = 1$ if $j = 33 - b$, and both are 0 otherwise. At some later clock cycle $j = J$ after all of the bits have trickled through the device, the output frequency and phase will thus be

$$f(t_J) \;=\; \sum_{b=1}^{32} \delta_b(f_2)\; 2^{b-1}\; f_0 = f_2 \qquad (4.5)$$

$$\begin{aligned} \phi(t_J) &= \phi(t_0) + \frac{2\pi}{f_{\rm clk}} \sum_{b=1}^{32} [J - (32 - b)]\;\; \delta_b(f_2)\; 2^{b-1}\; f_0 \\ &= \phi(t_0) + \frac{2\pi}{f_{\rm clk}} \left[J f_2 + \sum_{b=1}^{32} -(32 - b)\; \delta_b(f_2)\; 2^{b-1}\; f_0 \right] \qquad (4.6) \end{aligned}$$

Since we are interested only in the net phase shift due to the variable delay, we can remove the initial phase and the expected term proportional to $J f_2$, leaving

$$\Delta\phi(f_2) = -\frac{2\pi}{f_{\rm clk}} \sum_{b=1}^{32} (32 - b)\; \delta_b(f_2)\; 2^{b-1}\; f_0 \qquad (4.7)$$

So, due to the propagation delay each bit produces a net phase shift

$$\begin{aligned} \Delta\phi_b &= -\frac{2\pi}{f_{\rm clk}} (32 - b)\; 2^{b-1}\; f_0 \\ &= -2\pi\; (32 - b)\; 2^{b-33} \qquad (4.8) \end{aligned}$$

and the total net phase shift is simply a sum of all $\Delta\phi_b$ for each bit b that is on in the set of frequency control bits that produce output frequency f_2. Note that in the time-reversed case when f_2 is zero and f_1 is non-zero, the net phase shift has the exact expression but with opposite sign. As a result, we can derive a general expression for the net phase shift due to the bit-wise propagation delay when the synthesizer switches from frequency f_1 to f_2.

$$\Delta\phi \;=\; 2\pi \sum_{b=1}^{32} [\Delta\phi(f_2) - \Delta\phi(f_1)]$$

$$
\begin{aligned}
&= 2\pi \sum_{b=1}^{32} \left[\Delta\phi_b \ \delta_b(f_2) - \Delta\phi_b \ \delta_b(f_1)\right] \\
&= 2\pi \sum_{b=1}^{32} -(32-b)\ 2^{b-33} \left[\delta_b(f_2) - \delta_b(f_1)\right] \qquad (4.9)
\end{aligned}
$$

To verify this result we apply this correction to the data shown in Figure 4.2. Note that the corrected data (solid circles) show no sign of the discontinuous phase change jumps. According to the general first-order delay model in equation (4.2), we fit these corrected data to a second-order polynomial. From these fit results $|\alpha| \leq 2.9 \times 10^{-18}$ rad/Hz, we can assign an upper limit for the non-linear phase delay of 0.3 mrad for worst-case frequency changes of $\Delta f = 8$ MHz centered around 12.6 MHz. For the high-sensitivity interferometers, this phase shift would correspond to a systematic error of ≤ 0.4 ppb. We have taken similar data for different frequency ranges, particularly around the center frequency of 12.631 770 MHz used in the experiment, and verified that the model in equation (4.9) always removes the phase jumps. As this test closely simulates what the atoms experience during the actual interferometer experiment, it assigns an upper limit to the possible systematic phase shifts from the DDS itself and any dispersive elements between the synthesizer and the phase detector.

The DDS unit used in the previous version of this experiment, model DDS-1 also from *Sciteq Communications*, had a clock frequency of 25 MHz and an internal anti-aliasing filter with a high-frequency rolloff starting at $\sim$10 MHz. This synthesizer's output, centered at 6.315 885 MHz, was filtered, doubled, and filtered again using external rf elements. We measured the propagation delay through these external rf elements. Over the range from 8.4 to 17 MHz, the delay varied by as much as 1.9 ns. This variation alone, which does not include the response of the anti-aliasing filter internal to the DDS, corresponds to a correction of $\sim$80ppb for high-sensitivity interferometers.

4.2 RF amplitude-dependent phase shifts

As described in Section 3.6.1, we construct the adiabatic transfer pulses by varying the rf amplitude to two acousto-optic modulators: the shaping AOMs. We vary the rf amplitude by passing a fixed amplitude 40 MHz signal through two separate voltage-controlled variable rf attenuators shown in Figure 3.22, where the voltage control is provided by the computer. The variable rf attenuators are fundamentally rf diodes that are forward biased by the control voltage. If the bias voltage is zero, the diodes do not conduct and present a large impedance to an incident rf signal. As the bias voltage increases above zero, the effective impedance seen by the rf signal shrinks to arbitrarily small values. Relative to the input rf signal, this impedance both reduces the amplitude and shifts the phase of the output rf signal. As a function of input control voltage, the transmission and the phase shift through the variable rf attenuator traces out the current-voltage curve of the diode. As discussed in Section 3.6, because the amplitude curve is strongly non-linear we must correct the control voltage in order to produce a linear response in the output rf amplitude. Here, however, we are concerned about the phase shift. Because this rf amplitude-dependent phase shift is present on the rf signal launched into the crystals of the shaping AOMs, it will also be transfered to the interferometer laser beams. In their current configuration (see Figure 3.9), the two shaping AOMs defract the light in opposite directions, one with the direction of the propagating sound wave and one against the sound wave. Thus, if the rf signal to the $F=3$ shaping AOM changes by phase $\Delta\phi(\mathrm{ON}-\mathrm{OFF}) = \phi_3$ from on to off, the optical phase of the $F=3$ beam will change by $+\phi_3$. On the other hand, if the rf signal to the $F=4$ shaping AOM changes by phase $\Delta\phi(\mathrm{ON}-\mathrm{OFF}) = \phi_4$, the optical phase of the $F=4$ beam will change in the opposite direction by $-\phi_4$. Thus, the phase difference between the Raman beams which is what the atom interferometers sense would change by $\phi_3 - (-\phi_4) = \phi_3 + \phi_4$. So, even if the phase change were exactly the same for each shaping AOM ($\phi_3 = \phi_4$), the effect would not cancel; it would add.

To reduce this rf-amplitude dependent phase shift, we introduce a "phase shifter" into the rf path for the $F=3$ and $F=4$ shaping AOM signals. These phase shifters are also rf attenuators, except they are chosen because they have a larger phase

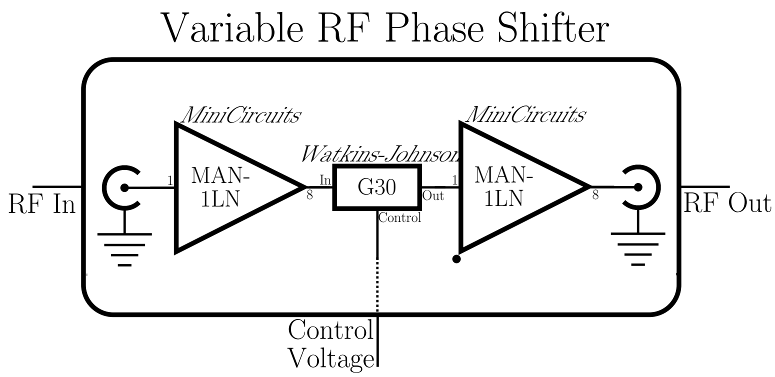


Figure 4.3: **RF phase shifter** used to correct for the phase variation of the variable radio frequency (rf) attenuators shown in Figure 3.22. The phase shifter consists of two rf amplifiers and a *Watkins Johnson* model G30 variable rf attenuator. A single supply operational amplifier (not shown) buffers the input control voltage to provide the bias for the G30. The G30 was chosen because it has an especially large propagation phase change over its full range of attenuation. By saturating the amplifiers and operating the G30 in a bias regime where its attenuation is relatively flat, the change in attenuation through the whole device can be reduced to less than 0.3 dB whereas the phase changes by more than 2 rad.

change for the same change in attenuation. We use the phase shifters to counteract the phase variation of the original variable rf attenuators. In a steady-state limit, there is a one-to-one mapping between the 12-bit digital value stored in the AWFG board memory and the phase shift that the rf attenuators will apply. As described in Section 3.6.1 the 12-bit integer is converted to an analog voltage by the AWFG board. This analog voltage is used to bias the rf diodes inside the rf attenuator. For this particular voltage value, the attenuators will attenuate and phase shift the input signal by a certain amount. Now for the phase shifter, there is a particular bias voltage that causes it to shift the phase of the incoming rf signal by an amount equal and opposite from that due to the variable rf attenuator. Thus, the phase shift from the variable attenuator can be compensated for by a particular bias voltage for the phase shifter. In the same way we constructed the numerical transform $p_a(x)$ to compensate for the non-linear amplitude response of the variable rf attenuator (see Section 3.6.1), we can construct another transform $p_\phi(x)$ to compensate for its phase shift.

For each of the two channels, the phase compensation system includes an additional 12-bit DA converter and two 128×8-bit flash memory chips which store the conversion waveform $p_\phi(x)$. The system's inputs are a copy of the AWFG board's internal sample clock and a copy of the 12-bit binary values x_3 and x_4 which the AWFG board converts to an analog voltage for the $F=3$ and $F=4$ output channels. For the $F=3$ channel, for instance, on every sample clock edge, the phase compensator uses the 12-bits of x_3 to index the two 8-bit memory chips in parallel. The two 8-bit values from the contents of the two memory chips are combined into a single 12-bit value $p_\phi(x_3)$ and 4 additional binary values. The 12-bit output $p_\phi(x_3)$ from the two memory chips is then converted to an analog voltage ranging from -5 to $+5$ V. This analog voltage goes to the $F=3$ phase shifter and biases it for the particular phase value required to cancel the phase shift the $F=3$ variable rf attenuator applies when it is biased with the individual amplitude control voltage $p_{a3}(x_3)$. Note that even if the attenuation through the phase shifter also varies somewhat with its control voltage, we can compensate for this variation by slightly modifying the transforms $p_a(x)$. For this reason, the linearization of the amplitude was performed after installing the phase shifters.

To determine the transforms $p_\phi(x)$, we sample the phase of the variable rf attenuators $\phi_a(x)$ when its control voltage $v(x = 0 \ldots 4095)$ is ramped linearly. We also determine the same curve $\phi_\phi(x)$ for the phase shifter. By inverting this last function we determine the digital control value $X(\phi) = \phi_\phi^{-1}$ which produces the phase shift ϕ at the phase shifter. The phase transformation then becomes $p_\phi(x) = X(\phi_a(x))$ for each possible value of x. To verify that this is in fact the correct transformation, we perform the same measurement but instead of a linear ramp for the phase shifter's control voltage we apply the transformed ramp $v(p_\phi(x = 0 \ldots 4095))$. This first iteration cancels most but not all of the phase from the variable attenuators. The phase error remaining after applying this transformed ramp we use to slightly correct the phase transform. We arrive at the final phase transformations $p_\phi(x)$ after no more than two iterations. Figure 4.4 shows the corrected and uncorrected phase response. The peak-to-peak variation of the corrected phase for each channel is less than 100 mrad, or more than an order of magnitude better than the ~ 1 rad peak-to-peak uncorrected phase change.

This phase compensation system functions in a feed-forward manner in the sense

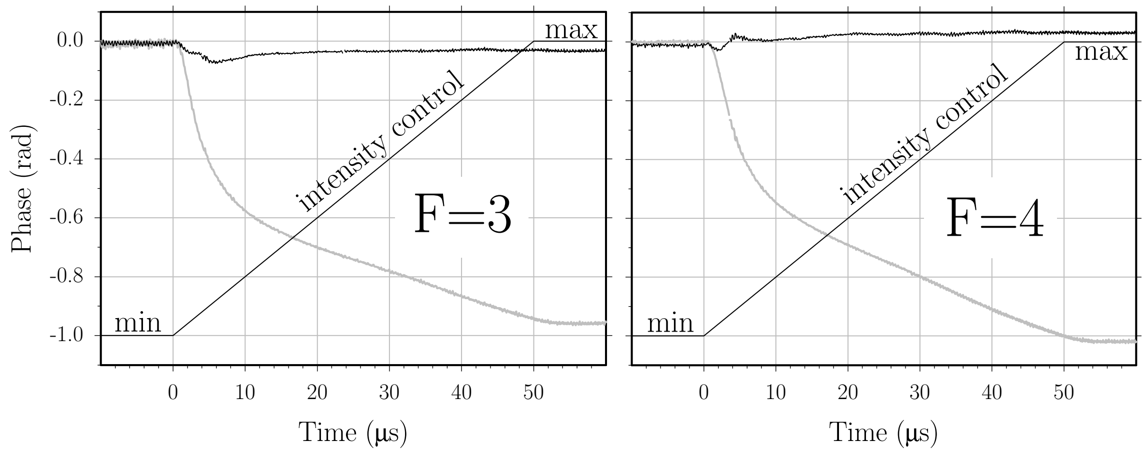


Figure 4.4: **Correction of the rf-amplitude dependent phase shifts from the variable rf attenuators.** Over 50 μs, the variable attenuators driving the $F=3$ and $F=4$ shaping AOMs are ramped from maximum (min) to minimum (max) attenuation. The resulting phase of the rf output relative to a fixed 40 MHz reference is shown in thick gray. To reduce this phase variation, we insert an rf phase shifter before each attenuator. We map out the phase versus control voltage response of these phase shifters and invert the resulting functions. Then, with the uncorrected phase response of the attenuators as an input, we apply the phase shifter transforms and determine the control voltages required to make the phase shifters exactly cancel the phase response of the attenuators. The variations with this feed forward system in place are shown in thin black. The corrected peak-to-peak variations are over an order of magnitude smaller than the uncorrected responses.

that it does not detect the phase change but knows in advance what the phase change will be. This works well as long as there is a one-to-one correspondence between the digital control value x and the amplitude and phase values resulting when the control voltages $v_a(p_a(x))$ and $v_\phi(p_\phi(x))$ are applied to the variable rf attenuator and phase shifter, respectively. In steady-state this is always true, but dynamically if the amplitude control voltage $v_a(p_a(x))$ changes slightly differently than does the phase control voltage $v_\phi(p_\phi(x)$, there will be a transient glitch as the phase shifter fails to precisely cancel the phase shift from the variable attenuator. To minimize this effect, we also tested the phase compensation system on the fastest possible ramp, when the value x changes from 0 to 4095 in one sample clock cycle. By tuning the delay and the time constant of the phase control voltage channels, we were able to reduce the glitch to the same size or smaller than the 100 mrad remnant phase deviation left over from the steady-state solution shown in Figure 4.4.

4.3 Beam collimation

The dependence on the Raman beam collimation is discussed in Section 6.1.1. By taking more data and using a better collimation tester, we reduced the uncertainty and eliminated a possible systematic error in our determination of the location of the Gaussian focus of both beams. Originally, we used a shear-plate collimation tester model 09SPM003 from *Melles Griot*. An incoming beam incident at 45° will reflect off both surfaces of the shear plate. The interference pattern from these two reflections can then be observed on a screen. In the direction of the shear, because the thickness of the glass plate varies, the propagation path difference and thus the phase between the two reflections changes across the beam. If the beam is exactly collimated, the resulting fringes will appear exactly orthogonal to the shear direction. If the beam is not collimated, the fringes appear to rotate about this direction, in one direction for a diverging beam and in the other direction for a converging beam. This collimation tester is extremely easy to use, because it produces a signal that is linearly proportional to the deviation from collimation. However, its accuracy is limited by how accurately the reference line is aligned to the shear direction of the plate. If the reference is off or if the angle of incidence of the incoming beam is slightly wrong, the collimation point can be systematically wrong.

A collimation tester with fewer potential systematic errors incorporates a parallel-plate instead of a shear-plate. Just as with the shear-plate, the interference between the reflection from both surfaces of the parallel-plate are observed on a screen. Unlike the shear-plate, however, the fringes from this interference pattern do not appear as lines. For an uncollimated incident beam, they appear as rings. As the beam approaches collimation, these rings expand until one central fringe occupies the entire region of overlap when the beam is perfectly collimated. If the central spot is always dark, one can readily collimate the beam by minimizing the amount of light visible at the screen. However, because this central spot can vary from bright to dark when the propagation path difference between the two reflections changes by as little as half an optical wavelength, in practice it is very difficult to keep the central spot dark. One common solution is to dither the angle of incidence of the plate slowly back and forth.

Since the propagation path difference depends sensitively on the angle of incidence, the interference phase difference shifts by several cycles. When the interference spot appears to flash on and off completely, the beam is collimated.

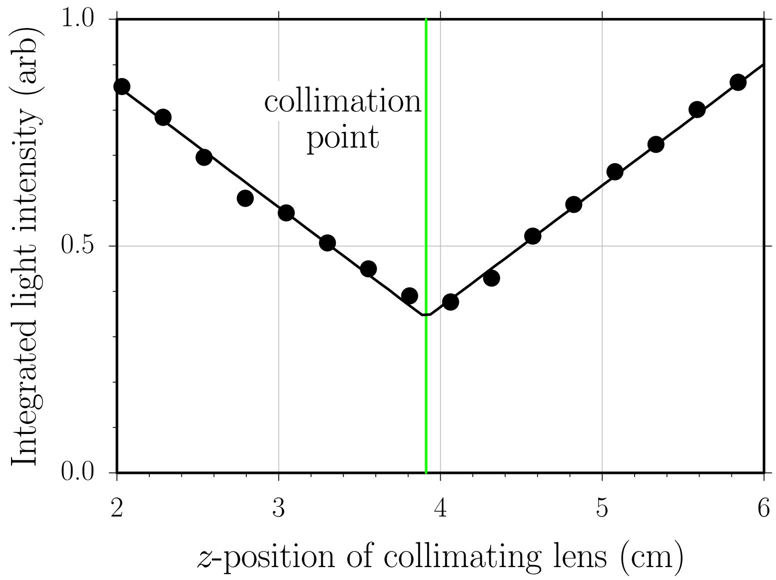


Figure 4.5: **Collimation of the bottom Raman beam.** A four inch diameter 1.6 cm thick precision parallel plate is placed in the bottom Raman beam above the vacuum chamber at an angle of incidence of roughly 45°. The incident Raman beam reflects off of both of the plate's uncoated surfaces onto a white screen placed to one side that is imaged by a video camera. Where the two reflections overlap, they interfere and produce fringes whose spacing is linearly proportional to how parallel the laser's wavefronts are at the plate. A piezoelectric transducer (PZT) slightly varies the incidence angle of the plate and thus adds a controllable propagation phase difference between the two reflections. By controlling the voltage across this PZT, one can fix the phase difference between the reflections so that the center fringe is dark. If the laser's incident wavefronts are parallel, the entire overlap region will consist of one dark fringe and the integrated light intensity will be minimized. As the wavefronts acquire some curvature, however, the spacing of the interference fringes shrinks and the minimum achievable light intensity increases. By recording the integrated video signal for different longitudinal positions of the collimating lens and fitting these data with the function $A|z - z_0| + B$, we can determine the lens position z_0 that collimates the beam to within a statistical limit of ± 0.014 cm, which corresponds to a lower limit for the radius of curvature of $\sim$27 km.

The method we use is slightly different, requires less operator skill, and is more repeatable. We image the screen with a video camera and observe the video signal on an oscilloscope. After capturing the entire screen of the video signal, we can integrate to obtain a numerical value for the total light incident on the screen. By controlling the incidence angle with a piezo-electric actuator, we can manually constrain the interference phase so that the central fringe stays dark long enough for us to capture the signal. We move the collimating lens toward and away from the source and at each step, we fix the lens, and acquire a total integrated light signal. As the lens moves away from the collimation point in either direction, the fringes become smaller and the minimum integrated light signal increases. The data from the bottom collimating lenses is shown in Figure 4.5. Fitting these data by an absolute value function, we obtain the position of the lens which minimizes the integrated light to within a statistical limit of ± 0.14 mm. The top collimating lens is aligned in the same manner to a similar limit of ± 0.17 mm. As a conservative upper limit, we estimated that both lens are positioned to within ± 0.2 mm, which corresponds to an uncertainty in the position of the Gaussian focus of 6.2 m and a maximum inverse radius of curvature of $5.2 \times 10^{-5}\,\mathrm{m}^{-1}$.

This collimation procedure is systematically limited only by the flatness of the parallel-plate. Our 4-inch diameter parallel plate comes from a collimation tester originally sold by *Blue Sky Research*. Although, this company no longer makes beam collimators, they referred us to the original optics designer who made the parallel plates [49]. He estimates that it is flat to better than $0.5\,\mu$rad. Using the above method to collimate a 1 cm Gaussian radius beam with a $0.5\,\mu$rad wedged plate would set the local radius of curvature to $\sim (1\,\mathrm{cm})/(0.5\,\mu\mathrm{rad}) = 20\,000$ m, approximately the same as our statistical uncertainty.

4.4 Relative angular alignment of beams

The photon recoil measured in this experiment derives fundamentally from the stimulated absorption of an optical photon with wavevector $\mathbf{k}_1$ from one laser and the stimulated emission of another optical photon with wavevector $\mathbf{k}_2$ into another independent laser beam. The first process is absorption and the second process is emission, so the effective wavevector determining the net momentum recoil of the atom is the vector difference of the two wavevector

$$\mathbf{k}_{\mathrm{eff}} = (\mathbf{k}_1 - \mathbf{k}_2) \tag{4.10}$$

Choose a coordinate system so that $\mathbf{k}_1 = k_1(0,0,1)$. Ideally, the second laser beam exactly counterpropagates with respect to the first $\mathbf{k}_2 = -k_2(0,0,1)$, so that the size of $\mathbf{k}_{\text{eff}}$ is

$$\begin{aligned} |\mathbf{k}_{\text{eff}}| = k_{\text{eff}} &= |\mathbf{k}_1 - \mathbf{k}_2| \\ &= |k_1(0,0,1) - [-k_2(0,0,1)]| \\ &= |(0,0,k_1+k_2)| \\ &= (k_1+k_2) \end{aligned} \tag{4.11}$$

If, however, the second laser is slightly misaligned from the first such that $\mathbf{k}_2 = -k_2(\sin\theta, 0, \cos\theta)$, equation (4.11) becomes

$$\begin{aligned} k'_{\text{eff}} &= |k_1(0,0,1) + k_2(\sin\theta, 0, \cos\theta)| \\ &= |(k2\sin\theta, 0, k_1 + k_2\cos\theta)| \\ &= (k_1^2 + 2k_1k_2\cos\theta + k_2^2)^{1/2} \end{aligned} \tag{4.12}$$

Because the recoil frequency is proportional to k_{eff}^2, the fractional error in the recoil measurement will be

$$\begin{aligned} (k'^2_{\text{eff}} - k^2_{\text{eff}})/k^2_{\text{eff}} &= \left[(k_1^2 + 2k_1k_2\cos\theta + k_2 + 2) - (k_1+k_2)^2\right]/(k_1+k_2)^2 \\ &= 2k_1k_2[\cos\theta - 1]/(k_1+k_2)^2 \\ &\simeq 2k_1k_2[(1-\theta^2/2) - 1]/(k_1+k_2)^2 \\ &= 2k_1k_2[-\theta^2/2]/(k_1+k_2)^2 \\ &\simeq -k^2\theta^2/(2k)^2 \\ &= -\theta^2/4 \end{aligned} \tag{4.13}$$

where the first approximation assumes $\theta \ll 1$ and the second approximation assumes that $k_1 \simeq k_2 = k$. Thus, a beam misalignment of only 63 μrad will shift our measurement of the recoil frequency down from the actual value by 1 ppb.

To guarantee that the beams are collinear to this limit or better, we align one beam to the other using pinholes. A simplified diagram of the beam path and relevant optics are shown in Figure 3.11. Each beam emerges from its fiber, is focused by a microscope objective to a Gaussian beam diameter of $114\,\mu$m, near which it passes through a pinhole of diameter $343\,\mu$m. It then freely expands to a diameter of $\sim$2 cm before it is collimated by a 2 m focal length lens. Between the two collimating lenses the beams overlap at the atoms. The two pinholes provide two reference points so that if it were not for the lenses, whenever both beams passed through both pinholes, they must be overlapped everywhere. With the lenses between the pinholes, however, this no longer holds true. Consider the case when the top pinhole is placed $f - \epsilon_{\mathrm{T}}$ before the top lens and the bottom pinhole is placed $f + \epsilon_{\mathrm{B}}$ after the bottom lens. In the limit of $\epsilon_{\mathrm{T}} \ll f$ and the distance between the two lenses $\simeq f$, after both lenses the image plane of the top pinhole is approximately $f + \epsilon_{\mathrm{T}}$ from the bottom lens. If $\epsilon_{\mathrm{B}} = \epsilon_{\mathrm{T}}$, the bottom pinhole will be in the image plane of the top pinhole. Consider now what happens when we control the input angle of the top beam to try to overlap it with the bottom beam which already passes through both pinholes. Because the two lenses image the pinholes on top of each other, no matter at what angle the top beam passes through the top pinhole it will always make it through the bottom pinhole. On first glance, if the pinholes happen to image each other, this alignment procedure seems to fail completely. However, it fails only if we ignore where the top beam intersects the top lens. If we can center the top beam on the top lens to within Δx, then we can constrain the input angle to roughly $< \Delta x/f$. Since the ratio of the angle after the lens to the input angle is ϵ_{T}/f, the angular deviation between the lenses should be $< \Delta x \epsilon_{\mathrm{T}}/f^2$. Since the confocal parameter for a beam with a Gaussian waist diameter of $114\,\mu$m is ~ 11 mm, it is reasonable to assume that $\epsilon_{\mathrm{T}} < 5$ mm. Assuming we can spatially overlap the beams between the lenses to within $\Delta x < 1$ mm, this limits the angular error from the lenses imaging the pinholes onto each other to $< (1\,\mathrm{mm})(5\,\mathrm{mm})/(2\,\mathrm{m})^2 = 2.5\mu$rad. However, to avoid this problem completely, we set the top pinhole as close as possible to the geometric focus of the top lens and the bottom pinhole over 30 mm away from the geometric focus of the bottom lens. To find the geometric focus of the top lens, we measure its focal length and then put the pinhole this exact distance away. To measure the focal length of the lens we measure the distance from the lens where two incident parallel 1 mm diameter beams from a helium-neon laser intersect. We find the point of intersection with a quadrant detector and then measure the distance to the detector using two calibrated rods of lengths 37.000 and 39.000 inches and a precision caliper. Since we know the lens is made from BK7 we can correct for the wavelength difference between He-Ne and our wavelength of 894.6 nm. With our calculated value for the focal length at 894.6 nm we again use the calibrated rods to set the pinhole this exact distance from the lens. Accumulating all of the possible measurement uncertainties, we estimate an error of ± 0.5 mm in positioning this pinhole at the focus. For the bottom pinhole, since we intentionally want to avoid the focus, we need only to use the diverging beam size to guarantee that we are well away from the geometric focus of the lens. With the bottom pinhole at least 30 mm away from the focus, even with our ~ 0.5 mm uncertainty in determining the correct position of the top pinhole, ϵ_{B} will always be much greater than ϵ_{T}, thus insuring that the pinholes never lie in each other's image plane.

In fact, the limit from this effect is smaller than the fundamental limit of our ability to center the beams on the pinhole. Because the top pinhole is so close to the focus of the top lens, the bottom beam will almost always also pass through it. Therefore, in order to make the beams parallel at the atoms, we need only direct the top beam backwards through the bottom pinhole. At 30 mm from a Gaussian focus diameter of $114\,\mu$m, the beam diameter is roughly the same as the pinhole diameter. We estimate that we can readily detect as small as a 5% drop in the power transmitted through the pinhole. By numerically integrating the transmission of a two-dimensional Gaussian through a circular aperture and inverting the resulting curve, we estimate that a tranmission of 95% corresponds to a relative beam-pinhole displacement of ~ 0.3 of the pinhole and beam diameter which is $\sim 120\,\mu$m, indicating an error of around $\pm 60\mu$m. Over the 2 m focal length of the lens, this corresponds to angular alignment error of $\sim \pm 30\,\mu$rad, which is a an error in the recoil frequency of -0.25 ppb, 64 times better than the previous limit with no pinhole.

4.5 Intensity matching

Effects which depend on the particular pulse shape may not cancel if the light level from the two optical frequencies and two possible beam directions are not all equal. Simulations indicate that if the difference frequency does not change to match the Doppler shift due to the gravitational acceleration during the $\pi/2$-pulses, the atoms will not be able to exactly follow the evolution of the dark state defined by the light. At the end of a $\pi/2$-pulse, the atoms will be in a state with a slightly different phase from the phase of the dark state. At the end of the interferometer the accumulated phase from this effect will change the final interferometer phase difference. Fortunately, the conjugate interferometer exhibits exactly the same phase shift and the effect cancels. However, an imbalance $\Delta I/I$ of the beam intensities can prevent this effect from canceling exactly. During the second and third $\pi/2$-pulses, the roles of the $F=3$ and $F=4$ light are opposite in the conjugate interferometer. If the amount of $F=3$ light emerging from one fiber is not the same as the amount of $F=4$ light emerging from the other fiber, the atomic state will slip away from the dark state by a different amount for an interferometer and its conjugate. Because the phase slip is not exactly the same, some of the effect remains after subtracting the final phase of the two interferometers. Simulations indicate that as much as 20 mrad of phase difference remains if the difference frequency is not chirped to compensate gravity and one of the four light levels ($F=3$ up, $F=4$ down, $F=3$ down, $F=4$ up) is 10% lower than the three others. For imbalances less than roughly 50%, the remaining uncanceled phase scales linearly with the fractional imbalance. Because the level of cancelation in the experiment may be even less than the simulation indicates, we attempt to avoid this problem by 1) chirping the difference frequency during the pulses to match gravity (see Section 3.6.3) and 2) balancing the beam intensities to better than 10%.

To balance the beam power levels, we measure the optical power in the center of each beam using an *EG&G Optoelectronics* model FND-100 photodiode placed just above the top window of the vacuum chamber. We program real interferometer pulse shapes into the synthesizers and run interferometer $\boxed{2}$ with no π-pulses. By measuring the light level at the end of the first and third $\pi/2$-pulse and at the beginning of the second and fourth $\pi/2$-pulse, we can directly observe the light levels for the $F=3$ and the $F=4$ emerging from one of the fibers. At these times during the $\pi/2$-pulses, both the $F=3$ and $F=4$ beams are on at the same time. The final phase of the interferometer is most sensitive to deviations from the dark state during these times, because at these points in the interferometers the dark state is a coherent superposition of two pure states. To measure the remaining two light levels, we flip the photodiode to look at the other beam coming from the opposite direction. We make slight adjustments to the $\lambda/2$-plate before the polarizing $F=3/F=4$ power splitter and then use two of the variable switchyard controls to maximize the light level when all of the four levels are equal to within $\sim 3\%$.

At early stages of this experiment, before the beam intensity balance was controlled, we estimate that the imbalance could have been as high as 30%. By measuring the beam balance several times during a long data taking run, we observed that the beam balance drifts by as much as 10% over many hours. In addition to random drift, because the atoms are inside the vacuum chamber and we are measuring the light outside the chamber on the other side of the top window, any transmission loss through this window will result in a systematic imbalance between the top and bottom beams. However, because this window is anti-reflection coated, it's total transmission loss will be less than $\sim 1\%$. Since this systematic imbalance is much less than the peak-to-peak random drift, we did not attempt to compensate for it. Also, since we are chirping the difference frequency to keep the atoms always in resonance, an additional effect from any remaining beam imbalance should be further reduced.

4.6 Crystal filters

All of the precision frequencies in the experiment are derived from the LORAN C reference signal at 10 MHz. This sinusoidal signal is passively quadrupled to generate 40 MHz, which is used to generate the 20 MHz TTL signal which replaces the internal clock of the AWFG board. A copy of this 40 MHz signal is passively doubled again to generate 80 MHz used to drive the $F=3$ AOMs of the switchyard. A copy of the

80 MHz is mixed with another copy of the 40 MHz signal to generate the 120 MHz for the common switch AOM. A 100 MHz VCO is phaselocked directly to the 10 MHz reference by dividing its output frequency by 10. This 100 MHz signal is mixed with another copy of the 80 MHz signal to generate the 180 MHz reference for the tracer phaselock. Since all of these reference frequencies are based on a multiplication of the original 10 MHz reference, they all have some remnant amplitude modulation sidebands at integer multiples of 10 MHz. This amplitude modulation is transferred directly to the lasers via any of the AOMs which diffract the beams. As discussed in Section 6.2.6, whenever the DDS's output frequency tunes near one of these sidebands, the two frequencies can mix down and add phase noise to the laser light at frequencies that affect the transfer efficiency and possibly systematically alter the phase of the interferometers. To minimize this affect we avoid tuning the DDS close to these "bad frequencies". As an addition precaution, we have installed crystal filters on the 40, 80, and 180 MHz reference frequencies. These filters have resonances that are typically less than 100 kHz wide, so all sidebands offset by integer multiples of 10 MHz are significantly reduced.

4.7 Dynamic response of the Raman beam AOMs

As shown in Figure 4.6 when the 40 MHz signal going to the radio-frequency (rf) amplifier driving the $F = 4$ shaping AOM is switched on rapidly, the diffracted light intensity fluctuates repeatably in time. These ripples did not depend on where the incident light beam passed through the AOM crystal. And, when we replaced the model P300AM-33 *TronTech* amplifier with an *IntraAction* PA-1264 better designed for switching, the fluctuations went away. Because the $F = 3$ shaping AOM exhibited the same behavior, its *TronTech* amplifier was also replaced with one from *IntraAction*. Although they do not shape the adiabatic transfer pulses, we also verified that the switchyard AOMs did not produce this intensity ripple.

While it is not clear whether these repeatable intensity fluctuations will systematically shift the interferometer phase, we swapped the amplifiers in and out while taking data and found that it did *not* significantly reduce the systematic phase shift

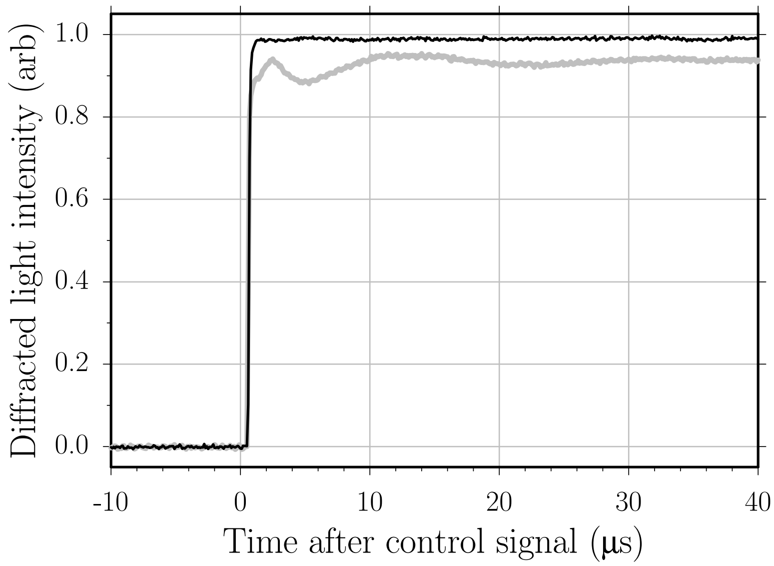
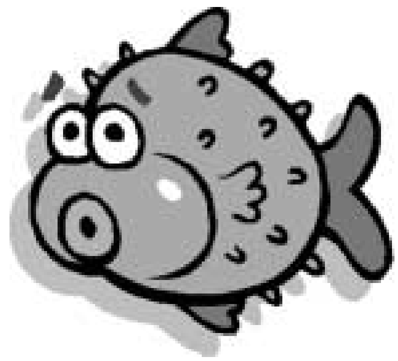


Figure 4.6: **Switching behavior of the $F=4$ shaping AOM.** The intensity of the diffracted order when the acousto-optic modulator is driven with the original radio-frequency (rf) amplifier (wide gray line) exhibits $\sim 7\%$ peak-to-peak fluctuations on the time scale of several μs. As shown by the thin dark line, when just the driving rf amplifier is replaced with one better designed for switching, the fluctuations vanish.

from the $\pi/2$-pulses discussed in Section 6.7.

Chapter 5

Results



5.1 Interferometer data

Using the accepted values for $\hbar/m_{\mathrm{Cs}}$ and $\lambda(\mathrm{CsD1})$ we can calculate an "accepted" value for the recoil frequency f_{rec} defined in Section 1. As other more precise measurements are made this accepted value will change. Since it allows us to more easily compare data taken at different times, we do not update our value of f_{rec} to match the current accepted value. Instead, we fix the lab value of the recoil frequency to a somewhat arbitrarily chosen value[1] of $f_{\mathrm{fix}} = 15\,006.278\,875$ Hz, which differs from the real value by less than several hundred parts per billion (ppb). It is of course the goal of this experiment to determine exactly how much this value differs from the real value. In this work, recoil data will always be presented as a difference from this fixed value, $\Delta f_{\mathrm{rec}} = f_{\mathrm{rec}}^{\mathrm{measured}} - f_{\mathrm{fix}}$ in units of Hz. Or, more commonly in units of ppb, $\Delta f_{\mathrm{rec}} = (f_{\mathrm{rec}}^{\mathrm{measured}} - f_{\mathrm{fix}})/f_{\mathrm{fix}} \times 10^9 = (f_{\mathrm{rec}}^{\mathrm{measured}}/f_{\mathrm{fix}} - 1) \times 10^9$

In the lab, we use this fixed recoil frequency to program the frequencies of our local oscillator, which is the difference frequency between the two Raman lasers for each light pulse. We program the pulse shape and beam direction data for the desired interferometer geometry into the synthesizers. We collect the atoms in a MOT and launch them vertically. After the atoms enter the magnetically shielded region inside the vacuum chamber, we trigger the programmed interferometer pulse sequence. The

[1] In fact, this value is close to the best accepted value as of March, 1998.

pulses of light split and then recombine the atomic wavefunctions of the atoms. If the two arms of the interferometer recombine coherently, the number of atoms emerging in one of the ground states will vary according to the total phase difference between the two arms of the interferometer. As discussed in Section 2.1, this final phase difference is zero only if we have the truly correct value for the recoil frequency (and possibly the gravitational acceleration) programmed into our synthesizers. If we do not use the correct value, then a different number of atoms will emerge from the interferometer. In principle, when we detect the signal proportional to this number, we can deduce the interferometer phase difference. However, in order to make this conversion, we must know the contrast exactly, and we must also know that variations of the signal result only from the interferometer phase difference and not from fluctuations in detection efficiency or in the number of atoms launched from the MOT. To make the phase measurement independent of the contrast and hence much less sensitive to signal amplitude fluctuations, we must find some way to scan across an entire fringe period by adding a controllable amount to the interferometer phase difference. We accomplish this by changing the Raman laser difference frequency of the final two $\pi/2$-pulses by a small amount f_s. Because our lab-based frequency reference now oscillates at a slightly different frequency between the final two $\pi/2$-pulses, over the time T a phase shift $\phi = 2\pi f_s T$ will accrue. If before every launch we change the value of f_s by a fraction of the fringe spacing $1/T$, we can trace out one or more fringes from minimum to maximum value. Each data set for one interferometer geometry consists of an array of 51 points $[f_s(i), a(i)]$, where $f_s(i)$ is the offset from the calculated center frequency, $a(i)$ is the signal size at that offset frequency, and $i = 1, 2, \ldots, 51$. The frequency span is always chosen to include exactly five fringes: $f_s(51) - f_s(1) = 5/T$. A non-linear least-squares fit routine (see Appendix C.2 for fit program) using the Levenberg-Marquardt method of root finding [50] fits the data $[\phi(i) = 2\pi T f_s(i), a(i)]$ by the function

$$f_{\text{fit}}(\phi) = A[1 + C\sin(B\phi + D)] \tag{5.1}$$

where A, B, C, and D are the adjustable parameters of the fit. This function is a sinusoid of amplitude AC about offset A with phase scaling factor B and phase offset D. The contrast defined as $(a_{\max} - a_{\min})/(a_{\max} + a_{\min})$ is given simply by $((A + AC) - (A - AC))/((A + AC) + (A - AC)) = (2AC)/(2A) = C$. In the limit that we exactly know the fringe spacing $1/T$, fit parameter B should always be unity. When we allow the fit routine to adjust B, its value varies randomly by at most $\pm 2\%$ centered on 1. Since this fluctuation is larger than any possible experimental variation of the value T, we attribute it to the statistical limit of determining the oscillation period of a sinusoid using only a finite number of points. For this reason we set $B = 1$ and do not allow the fit routines to vary it[2]. The final parameter D is the measurement result. It represents the amount of phase or frequency we must shift our fixed value for the recoil frequency to arrive at the actual value.

Of course, because of its dependence on detunings and particularly the local gravitational acceleration, the parameter D emerging from a single interferometer is not enough to derive a sufficiently accurate value for the recoil shift (see equation (2.21)). At the very least we must also measure the phase from the conjugate interferometer geometry ($\boxed{2}$ with $\boxed{1}$ and $\boxed{4}$ with $\boxed{3}$) in order to get a single measurement of the recoil frequency. To get the best cancelation of the unwanted common dependencies, we should measure both interferometer geometries simultaneously. Unfortunately, because a dark-state transfer "$\pi/2$-pulse" cannot transfer atoms from a superposition state to both pure states simultaneously, only one interferometer geometry can be constructed at one time. As a compromise, however, at each point we alternate between interferometers. If we were taking data using interferometer geometries $\boxed{1}$ and $\boxed{2}$, for instance, for the first launch we would build $\boxed{1}$. For the next launch we would reprogram all of the synthesizers to build to interferometer $\boxed{2}$. For the third launch would reprogram the synthesizers back to $\boxed{1}$ and so on. After 102 launches, each time switching between interferometers $\boxed{1}$ and $\boxed{2}$, we would have 51 points from each interferometer taken at almost exactly the same time. Only fluctuations occurring on the time scale or faster than the launch repetition rate of 1/0.908 s, would not cancel. On the other hand, because it takes at least 10 launches to acquire a full fringe cycle, fluctuations on the scale of $\sim$1 s or faster would appear as noise

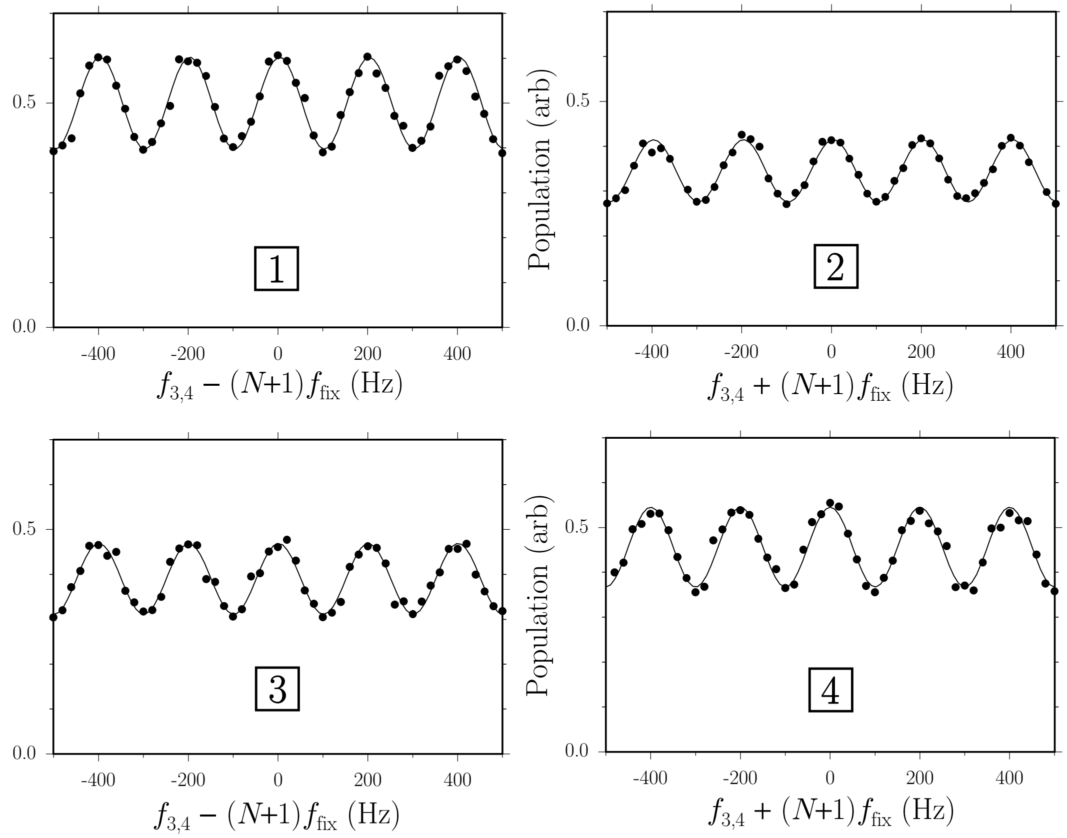
[2]The possibility that an uncertainty in the value of T contributes to a systematic error is discussed in Section 6.6.3.

on the fringes and should not change the final phase difference between the two interferometers. To achieve the best common-mode cancelation, all of our interferometer data are taken either by switching between $\boxed{1}$ and $\boxed{2}$ and then between $\boxed{3}$ and $\boxed{4}$ or by alternating between all four interferometers $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, $\boxed{4}$, $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, $\boxed{4}$, etc.

We now present the data from a single set of interferometer fringes for three representative configurations: $T = 5$ ms with 30 π-pulses, $T = 120$ ms with 30 π-pulses, and $T = 120$ ms with no π-pulses. In each case the data appear as we see them on the computer screen. The vertical axis is the integrated signal from the photomultiplier tube (PMT) that is proportional to the number of atoms in the $F\!=\!4$ state. Because of the magnetic sublevel sensitive transfer from the DF Raman laser discussed in Section 3.1.6, this $F\!=\!4$ signal is proportional to the number of atoms emerging from the interferometer in the $F\!=\!3, m_F\!=\!0$ state. It is scaled to the fit parameter A from equation (5.1) of interferometer $\boxed{1}$. The horizontal axis is the two-photon difference frequency offset of the third and fourth $\pi/2$-pulse from their center frequencies giving by $\pm(N\!+\!1)f_{\text{fix}}$, where the sign is determined by the interferometer geometry and N is the number of π-pulses. This axis spans exactly five fringes, or $5/T$.

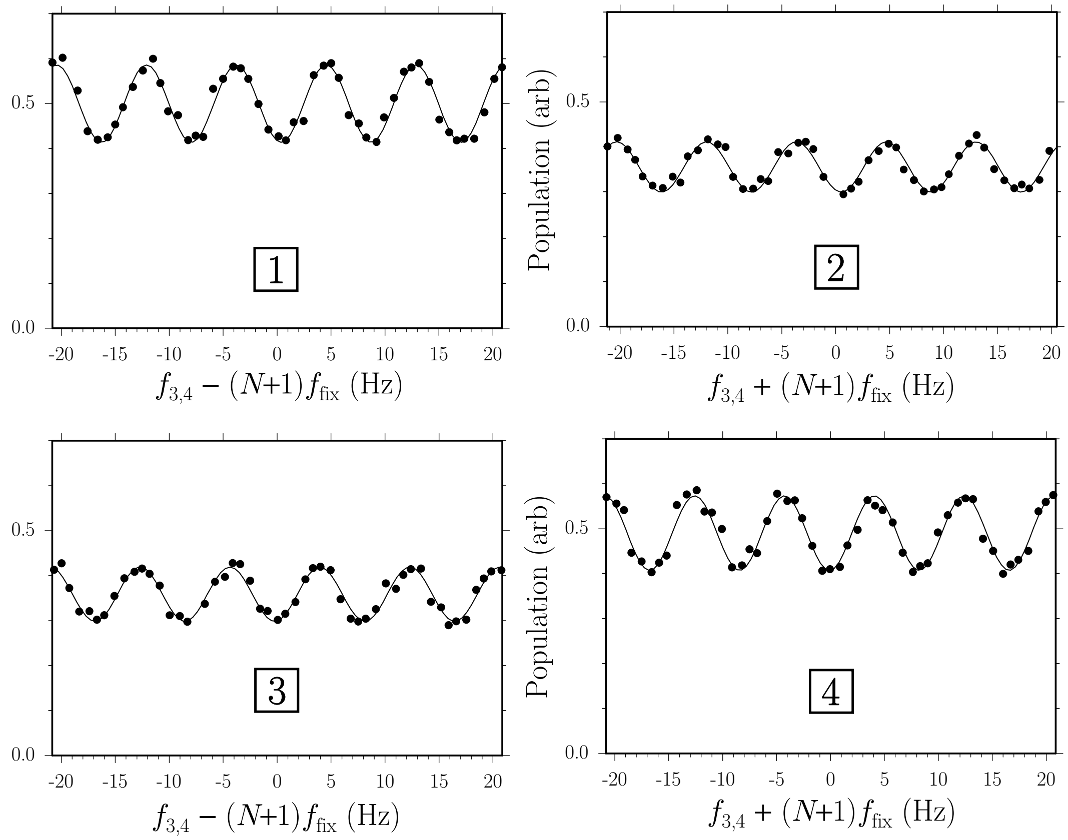
Below the fringes are the resulting fit parameters, A, C, and D from equation (5.1) and the value of χ^2. From the phase parameters D, we calculate the per recoil frequency shift $\Delta f = -\Delta D/[2\pi T(N+1)]$, which is the deviation from f_{fix} in Hz. Averaging these corrections Δf_{norm} and Δf_{inv} from the normal and inverted interferometers, respectively, gives the result from a complete measurement of f_{rec}, given in units of Hz and ppb relative to f_{fix}.

Figure 5.1: **Interferometer data for all four interferometers with $T = 5$ ms and $N = 30$ π-pulses.**



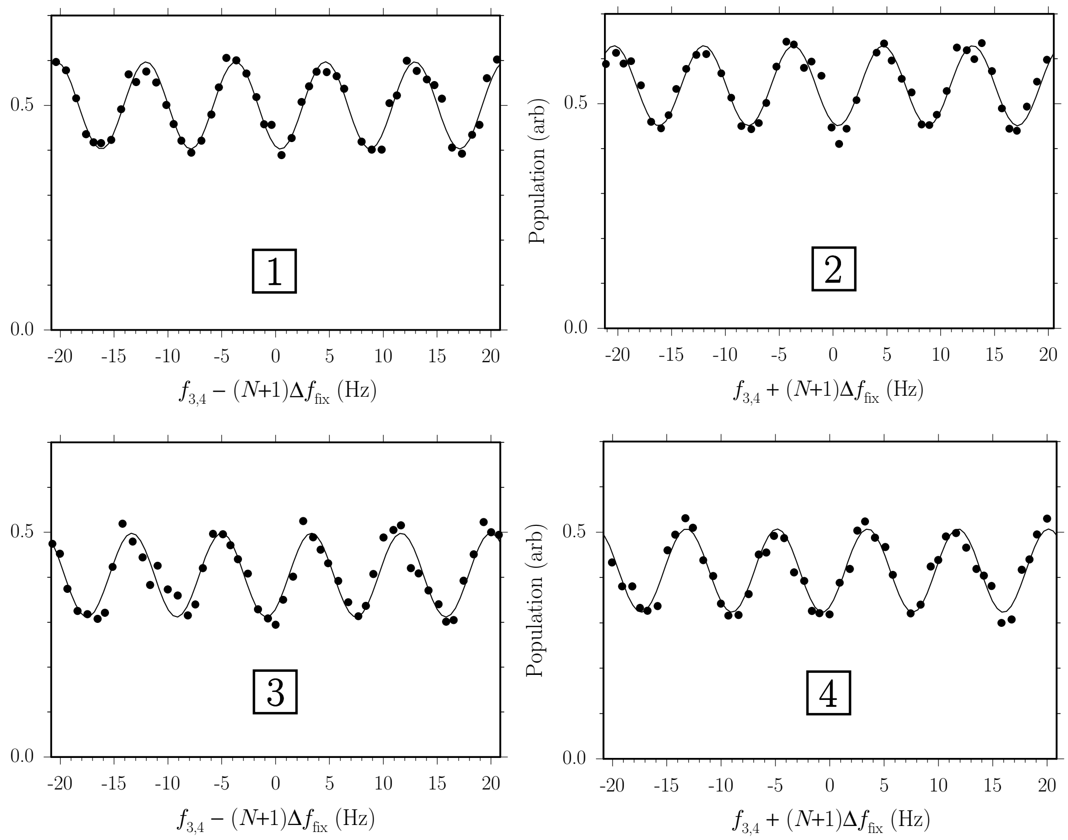
Fit Parameter	• **Interferometer Geometry** ○			
	1 Normal	2	3 Inverted	4
A (arb)	0.500 0(1 7)	0.342 0(1 7)	0.389 9(2 0)	0.456 2(2 0)
C (%)	20.49(47)	20.11(69)	20.23(74)	19.50(63)
D (rad)	+1.456 7(23 3)	+1.447 3(35 9)	+1.537 7(36 9)	+1.567 4(32 7)
χ^2	1.068×10^{-6}		1.583×10^{-6}	
Δf (Hz)	−0.009 7(44 0)		+0.030 6(50 6)	
6 $\frac{1}{2}(\Delta f_{\rm norm} + \Delta f_{\rm inv})$		= +0.010 5(33 5) Hz = +349(1 117) ppb in $f_{\rm fix}$		

Figure 5.2: **Interferometer data for all four interferometers with $T = 120$ ms and $N = 30$ π-pulses.**



Fit Parameter	• Interferometer Geometry ○			
	1 Normal 2		3 Inverted 4	
A (arb)	0.500 0(1 9)	0.355 3(1 9)	0.358 7(2 0)	0.490 0(2 0)
C (%)	17.22(53)	15.74(78)	16.83(79)	16.91(57)
D (rad)	−1.862 4(32 8)	−1.961 8(49 0)	−1.413 3(49 4)	−1.493 5(35 4)
χ^2	3.564×10^{-6}		3.959×10^{-6}	
Δf (Hz)	−0.004 2(2 5)		−0.003 4(2 6)	
6 $\frac{1}{2}(\Delta f_{\rm norm} + \Delta f_{\rm inv})$	$= -0.003\,8(1\,8)$ Hz $= -128(60)$ ppb in $f_{\rm fix}$			

Figure 5.3: **Interferometer data for all four interferometers with $T = 120$ ms and $N = 0$ π-pulses.**



Fit Parameter	• Interferometer Geometry ∘			
	1 Normal 2		3 Inverted 4	
A (arb)	0.500 0(2 7)	0.540 1(2 7)	0.404 6(3 5)	0.415 1(3 5)
C (%)	19.42(78)	16.51(69)	23.08(1.25)	22.15(1.22)
D (rad)	−1.916 4(38 5)	−1.934 2(44 5)	−0.931 9(53 4)	−1.054 1(53 9)
χ^2	1.345×10^{-5}		2.360×10^{-5}	
Δf (Hz)	−0.023 6(78 0)		−0.162 0(100 7)	
6 $\frac{1}{2}(\Delta f_{\text{norm}} + \Delta f_{\text{inv}})$	= −0.092 8(63 7) Hz = −3 093(2 122) ppb in f_{fix}			

We can use the results of Section (3.7.1) to estimate the full-width half-maximum (FWHM) of the resonance lineshape that these fringes are superimposed on top of. From Figure 3.33, the FWHM of 70 μs long full intensity π-pulses is approximately 253 kHz. Since the $\pi/2$-pulses are $250/70 = 3.57$ times longer and on average half as intense, according to equation (3.23), they should be approximately $(2(3.57))^{-\frac{1}{2}} = 0.374$ times as wide, or roughly 94.6 kHz. Even for the shortest interferometers $T = 5$ ms, the total span $5/T = 5/5\,\text{ms} = 1$ kHz is much smaller than this resonance linewidth, and thus the curvature at the top of the lineshape is invisible in the data.

In Figure 5.1 we have measured the separation of the resonances to within roughly 0.15 Hz and thus identified the center of each resonance to within approximately $1/\sqrt{(2)}(0.15\,\text{Hz})/(95\,\text{kHz}) = 1.1 \times 10^{-6}$. The same number for the other two cases are 5.9×10^{-7} from Figure 5.2 with $T = 120$ ms and 30 π-pulses and 6.7×10^{-7} from Figure 5.3 with $T = 120$ ms and no π-pulses.

5.2 Noise

Random fluctuations of the final signal ultimately limit the precision with which we can resolve the position of the fringes and determine the size of the recoil frequency. For interferometer fringe signals of the form given in equation (5.1), there are three general categories of noise: offset, contrast, and phase, corresponding to fluctuations of the parameters A, C, and D, respectively. *Offset* noise is a fluctuation of the mean value of the fringe signal that can be caused by variations in the number of atoms launched from the MOT or small changes in the detection efficiency. It could also be caused by variations in the efficiency of any of our adiabatic transfer π-pulses. Small amounts of offset noise ΔA contribute a phase uncertainty of roughly $\Delta\phi = \Delta A/C$, where C is the interferometer contrast. *Contrast* noise is a variation of parameter C. Fortunately, in the limit that the fringes can still be resolved, pure contrast noise does not contribute to the final phase uncertainty $\Delta\phi$. Finally, there is *phase* noise which contributes directly to uncertainty of the final signal.

As in Figure 5.2, the uncertainty ΔD of the phase fit parameter D after 51 launches using a sensitive interferometer is typically less than 50 mrad, which is equivalent to an rms phase noise of roughly $(50\,\text{mrad})\sqrt{51} \simeq 360$ mrad per launch.

Launch signal

As described in Section 3.1.5 the most basic launch signal comes from a single pulse of the probe laser. For magnetic sublevel sensitive detection (see Section 3.1.6), however, we must add at least one Doppler-free (DF) π-pulse from the DF Raman laser and a pulse from the clearing beam. The signal from a launch followed by pulses from the three detection beams typically fluctuates somewhere between 1.5% and 2.5%. By temporarily removing the DF Raman transitions, we can verify that the DF Raman transfer contributes roughly half, 1.1% to 1.8% of this root-mean-square (rms) offset noise.

Signal background

In addition to the 51 fringe data points, we also take 3 background points to establish a value for the true signal zero. To take a background point we detune the laser difference frequency of the last two $\pi/2$-pulses by $f_s = f_{\text{bkgnd}} = 1$ MHz which is much larger than the 100 kHz two-photon linewidth of these pulses and thus out of resonance with the atoms. Because the final two pulses do not address the atoms, the only remaining signal is due to either 1) remnant cesium atoms from the launch, 2) trace amount of cesium vapor always in the vacuum chamber, or 3) scattering of the probe beam into the photomultiplier tube.

By adding more π-pulses the signal can be made arbitrarily small, and consequently the relative size of the background signal varies greatly. For most of the data, however, this background is less than 10% of the signal level. The fractional standard deviation of 50 background points taken one after the other is typically less than 1.3%. If in the worst case this background is 30% of the signal, then background fluctuations will contribute no more than $0.3(1.3\%) = 0.4\%$ offset noise to the signal, which is negligible compared to the noise on the launch signal

Adiabatic transfer pulses

To investigate the amount of offset noise contributed by the adiabatic transfer π-pulses, we set up a launch sequence similar to the one described in Section 3.7.1: one velocity selecting π-pulse followed by N regular π-pulses, a clearing pulse, one DF Raman π-pulse, and finally the detection probe pulse. For $N >> 1$, small variations in the π-pulse efficiency can significantly vary the size of the final signal. With 30 π-pulses, the adiabatic transfer π-pulses typically add from 1.5% to 2.9% rms offset noise. This gives a total rms offset noise of 2.5% to 3.5%. With a worst case contrast of 16%, this typical offset noise of around 3.0% corresponds to a phase noise of $0.03/0.16 \simeq 190$ mrad, or just slightly over one third of the total phase noise.

The $\pi/2$-pulses indirectly contribute to the total noise. First, they establish the interference contrast which determines how sensitive the fitted phase uncertainty is to offset noise. And more importantly, they are the vehicle by which the phase noise of the lasers is transfered to the atomic wavefunctions. Motion of the laser wavefronts applied both electronically by the frequency and phase locks and mechanically by the motion of optics relative to the free falling atoms causes the interferometer phase difference to fluctuate from launch to launch. From the lock error signals we can set lower limits on the contributions from the Raman phase lock loop (PLL), the tracer PLL, and the vibration isolation (VI) system.

From the tracer and Raman PLL error signals we estimate lower limits of 6 mrad and 14 mrad, respectively, on the rms phase noise greater than roughly 100 Hz. For the VI system we take the closed loop error signal shown in Figure 3.17, apply the sensitivity function in equation 3.18, and integrate from 0.01 to 100 Hz. This gives a lower limit of 11 mrad on the rms phase noise due to motion of the interferometer platform holding the top polarizing beamsplitter (PBS) cube. Even summed in quadrature, these estimates are still too small to explain the $\sim$300 mrad not related to offset noise.

One quite reasonable possibility that might account for the missing noise would be if the *true* error signals from the various locks were significantly larger than the *lock* error signal. Additional sources of noise that might make the true error signal of the Raman PLL larger include 1) phase noise from the microwave reference or from the direct digital synthesizer (DDS), 2) noise from the detection and/or amplification of the microwave beatnote, and 3) instabilities of the 9 GHz voltage controlled oscillator (VCO) that are too fast for the Raman PLL, whose bandwidth is ~ 200 kHz, to correct.

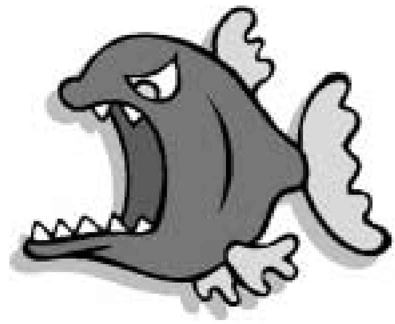
The most likely culprit for increasing the true error signal of the tracer PLL is the wavelength difference between the tracer and the Raman beams. Any motion or thermal variation of optics will cause an optical phase shift that depends on the laser's wavelength. Because the tracer and Raman laser wavelengths differ by $(2\,\text{nm})/(894.5\,\text{nm}) \simeq 2.2\%$, the tracer PLL can reduce the phase noise of the Raman beam by no more than this amount, even as the lock error signal goes to zero.

Another possible source of wavefront motion that the tracer PLL might not be able to completely remove could be air currents from the room's air conditioning system. As air of different temperatures and densities moves across the beams it changes the local index of refraction and thereby shifts the position of the laser wavefronts. To minimize this effect, over the entire path length between the fiber outputs and the vacuum chamber, plastic tubes and/or cardboard boxes are placed as close as possible to the beams to shield both the Raman and tracer beams from any air currents.

Finally, for the VI system, there are numerous reasons why the lock error signal may not represent the true motion of the interferometer platform. Besides the noise floor of the accelerometer, because the accelerometer is not located exactly at the top PBS cube, rotational motion such as the entire optical table or the VI support structure tilting will not be exactly removed by the feedback system. Similarly, if the accelerometer or the axis of the air bearing are misaligned from the Raman beam direction, the motion detected by the sensor will not exactly match the motion of the top PBS cube, and the feedback loop will not correctly cancel it out.

At separate times we have lowered the overall gain of each of these three feedback loops, and each of the loop gains could be lowered by over a factor of 8 before the fit uncertainty ΔD of the final interferometer phase increased noticeably. Given the magnitude of each of these integrated error signals, this indicates that either 1) the lock error signal is an accurate representation of the true error signal, or 2) the true error signal is roughly 8 times larger than the lock error signal.

We have investigated and improved each of these feedback systems and ultimately achieved only marginal improvement of the final phase uncertainty. From our general experience running this experiment, it is clear that this final uncertainty is not limited by a single source of noise. Thus, improving the signal-to-noise by even less than a factor of two would require redesigning and rebuilding a significant fraction of the experiment.



Chapter 6

Checks for systematic errors

Proving that a measurement is correct to a certain level is an unending and somewhat subjective process. One clearly inefficient approach is to turn every possible knob in the experiment and verify that the final result never changes. An alternate approach is to think of all of the possible ways we could be systematically making the wrong measurement and then either conduct tests for or convince ourselves through theoretical predictions that such each effect will be too small to make a difference. In the end, both approaches are limited by our ability and the ability of our colleagues to think of all of the possible knobs or different possible effects. Nevertheless, in this section we group the possible systematic effects we have considered into general categories, discuss each effect, and present the experimental and/or theoretical reasons for assigning a limit to how much it could affect the final measurement result.

The photon recoil experiment provides many handles for testing potential systematic effects. Within a few seconds, we can alter the geometry of the atom interferometer by changing the timing of the pulses, the number of π-pulses, the positions in the fountain trajectories where the light-atom interactions occur, the intensity and shape of the optical pulses, *etc.* We can also change the frequency offsets, polarization, alignment and wavefront curvature of the laser beams and vary environmental factors such as the magnetic bias field. For each of these variables we use experimental tests, theoretical predictions, and sometimes both to set limits on the possible measurement error these variables will produce.

One of the most important variables we have is the free evolution time T. This is the time between the first and second $\pi/2$-pulses and between the third and fourth $\pi/2$-pulses when the atoms evolve freely in a superposition of the hyperfine ground states. By varying T and using an analysis method that I developed, we can simultaneous measure and remove any fixed phase error from the $\pi/2$-pulses. Because the phase error from the $\pi/2$-pulses was the largest remaining systematic error, this particular test above all others has made this measurement possible.

Many of the potential systematic effects are canceled because we always make two measurements, one using an interferometer geometry which pushes the atoms down and the other using the conjugate geometry which pushes the atoms up. As discussed in Section 1.1, a single measurement of the recoil shift is the difference of these two results, at which point many systematic errors subtract out. Also, we routinely change the recoil direction of all of the interferometer pulses, thereby interchanging the role of the up and down interferometers. The difference between the results from these inverted interferometers provides another measurement of the recoil shift. Averaging the results from the normal and inverted interferometers further reduces systematic problems such as those that arise from the gravity gradient and from magnetic field shifts.

The results of this section are summarized at the end in Tables 6.2 and 6.3. We find a total systematic correction of +2.74 ppb to f_{fix}, or $+82.23\,\mu$Hz. Summing the systematic uncertainties in quadrature gives a total systematic uncertainty of 3.23 ppb in the recoil frequency[1].

6.1 Beams

The optical wavefronts act as the "spatial ruler" against which the evolution of the atoms is compared. Consequently, our measurement is only as good as our ruler. Anything which bends, shifts, or in some way distorts the position of the wavefronts will affect the final measurement. In Section 2.1 where we discussed how the lasers and the atom's free evolution contribute to the final interferometer phase, we simplified

[1] Remember that the uncertainty in α is half as much, or 1.62 ppb.

the problem by describing the lasers as plane waves with a single wavenumber k_{eff}. Real lasers always have some finite spatial extent and propagate according to the wave equation derived from Maxwell's equations. The amplitude profile of the fundamental TEM00 mode of a real Gaussian beam with wavenumber $k = 2\pi/\lambda$ is described by

$$u(r,z) = \frac{w_0}{w(z)} \exp\left[-i\left(kz - \Phi(z)\right) - r^2\left(\frac{1}{w^2(z)} + i\frac{k}{2R(z)}\right)\right] \tag{6.1}$$

where

$$w^2(z) = w_0^2\left[1 + \left(\frac{\lambda z}{\pi w_0^2}\right)^2\right] \tag{6.2}$$

is the local $1/e$ beam radius,

$$R(z) = z\left[1 + \left(\frac{\pi w_0^2}{\lambda z}\right)^2\right] \tag{6.3}$$

is the local radius of curvature, and

$$\Phi(z) = \tan^{-1}\left(\frac{\lambda z}{\pi w_0^2}\right) \tag{6.4}$$

is the Guoy phase. Even located on axis ($r = 0$) exactly at the focus when $z = 0$ with no distortion, the wavefront spacing differs from $k = 2\pi/\lambda$ because of this last term. On axis, the local wavefront gradient will be modified by

$$\Delta k = -\frac{\partial \Phi}{\partial z} = \frac{-1}{1 + (z/z_0)^2}\frac{1}{z_0} \tag{6.5}$$

where

$$z_0 = \frac{\pi w_0^2}{\lambda} \tag{6.6}$$

is the confocal parameter. Evaluating equation (6.5) at the focus when $z = 0$, the fractional change in the wavenumber

$$\frac{\Delta k}{k} = -\left.\frac{\partial \Phi}{\partial z}\right|_{z=0}\left(\frac{1}{k}\right) = \left(\frac{-\lambda}{\pi w_0^2}\right)\left(\frac{\lambda}{2\pi}\right) = \frac{-\lambda^2}{2\pi^2 w_0^2} \tag{6.7}$$

For our laser beams with $\lambda = 894.60$ nm and $w_0 = 0.9565 \pm 0.074$ cm, we have $\Delta k/k = -0.4432 \pm 0.0069$ ppb, which requires us to correct the recoil frequency measurement by twice as much or $+0.886 \pm 0.014$ ppb. Besides this fundamental correction, there are many other ways in which the laser wavefronts we have in the lab differ from the ideal. In this section, we will discuss these differences and the issue of beam polarization.

6.1.1 Wavefront curvature

In Section 4.3 we discussed our improved method for positioning the collimating lenses so that the 2 cm diameter Gaussian focus of each beam is no more than $\sim$6 m away from the atoms. As one moves away from the focus of one of the beams, the magnitude of the local wavefront gradient changes as shown in Figure 6.1. The change is relatively small for movement along the axis of the beam compared to the much more rapid change as one moves radially outward from the beam center. In order to arrive at a reasonable upper limit for the maximum wavefront curvature change the atoms will experience, we must estimate how the atoms move with respect to the beam.

An early version of this experiment was severely limited by systematic wavefront distortion because the Raman beams were horizontal and as the atoms fell transversely across the wavefronts, they experienced large wavefront shifts [27, 51, 52]. To the extent that our beams are aligned vertically with gravity, the center of the atomic cloud does not move radially in the beam and therefore samples much less distortion.

Transverse motion

The atoms move transversely due to three possible misalignments: the initial launch direction, the alignment of one Raman beam relative to the other, and the alignment of both Raman beams to gravity. We attempt to launch the atoms as vertically as possible. We move the probe beam to center it on the atom cloud as the atoms make their first pass through the detection region (early probe) on their way upward. We then set the probe to flash on when the atoms pass through the detection region on

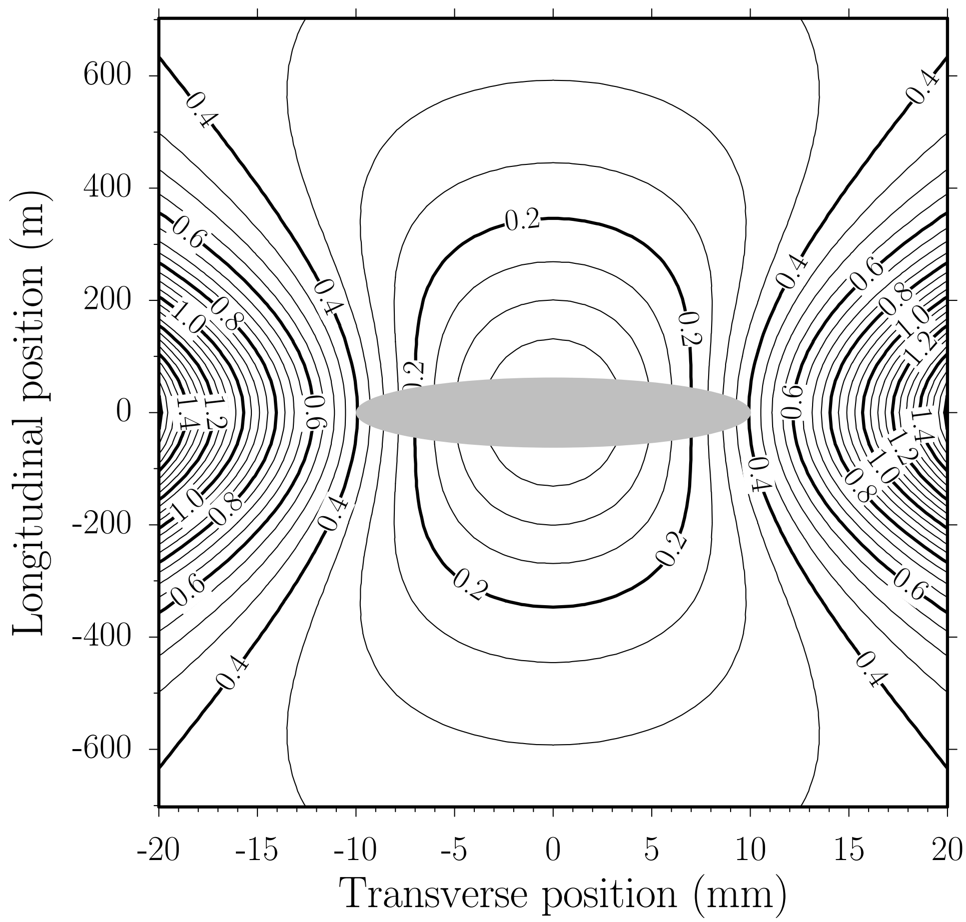


Figure 6.1: **Magnitude of the wavefront gradient for a $2w_0 = 2.0$ cm diameter Gaussian beam** in parts per billion (ppb) deviation from the gradient at the origin as a function of the longitudinal displacement from the focus and the transverse displacement from the beam axis. The gray ellipse at the origin indicates *ten times* the size of our uncertainty in locating the origin of the beam.

their way back down (late probe). We optimize all of the available adjustments for the fountain (see Section 3.1.4) to maximize this late probe signal. Although this signal depends on many factors, it guarantees that the atoms are not being launched to one side. From the accuracy with which we can use the early probe signal to overlap the probe beam with the atoms, we estimate that the angle θ_{L} between the initial launch direction and vertical is no more than 0.3 mrad. If the last interferometer pulse occurs no more than 430 ms after the launch, the center of the atom cloud will have shifted by $\Delta x_{\mathrm{L}} = v_{\mathrm{L}}\theta_{\mathrm{L}}(430\,\mathrm{ms}) < 0.4$ mm across the beam.

In addition to the initial launch direction, if the Raman beams are misaligned by an angle $\theta_{\rm B}$ with respect to each other, the velocity change $v_{\rm r}$ from each momentum recoil will have a non-zero transverse component. From equations (2.6), (2.23), and (2.39) with the assumption that $T' = NT_{\pi\pi}$, the difference in position at the last $\pi/2$-pulse for the up and down interferometer is

$$\Delta z = z_4(\mathrm{up}) - z_4(\mathrm{down}) = v_{\rm r}[2(N+1)T + NT'] \tag{6.8}$$

For $\theta_{\rm B} \ll 1$, the difference in position across the beam is $\Delta x = \Delta z \theta_{\rm B}$. In Section 6.1.4 we report a long-term mean measurement error of -1.0 ppb due to relative misalignment of the Raman beams. This error in the recoil shift corresponds to an angle of $\theta_{\rm B} = 63\,\mu$rad. In the absolute extreme case with $T = 160$ ms and 50 π-pulses $\Delta z = 12$ cm, so $\Delta x_{\rm B} = 12\,\mathrm{cm}(63\,\mu\mathrm{rad}) = 7.6\,\mu$m, which is small enough compared to the other effects to be neglected.

The Raman beams are aligned to gravity using an *Applied Geomechanics* 755-1129 tilt sensor. This dual axis tilt sensor repeatably detects rotational displacements of its case with respect to gravity with a precision better than 1 μrad. Onto the bottom of this sensor, we glued a high-quality 2 inch diameter dielectric mirror. The mirror plus tilt sensor rest on three stainless steel balls glued to a ring which is mounted in a 2 inch mirror mount. This setup allows us to position a downward facing reflective surface so that is normal to gravity. The three balls define a plane which can be tilted with respect to a fixed mount. Since the mirror rests on the three balls, the plane of the mirror will be parallel to the three-point plane, limited only by surface roughness of the mirror. By turning the mirror plus tilt sensor as a unit, we can rotate the mirror surface with respect to the three-point plane. Since the tilt sensor is attached rigidly to the mirror, if the three-point plane is not normal to gravity, the sensor will detect a tilt as the two planes rotate with respect to each other. In fact, if the outputs from the two orthogonal tilt sensors are plotted in an xy-plane, the readings will trace out a circle as the unit is rotated. The radius of this circle represents how much the three-point plane and mirror surface are tilted from perfectly horizontal. The coordinates of the center of the circle represent the fixed tilt offset between the mirror surface and the sensor. Once we have measured this fixed tilt offset, we no longer need to rotate the mirror. We simply use the mirror mount controls to tilt the mirror surface until the tilt sensor reproduces these fixed values. At this point, to the extent that the tilt sensor repeats, the mirror will be normal to gravity. By recording the values from six different orientations equally spaced around a circle and fitting the results with two sinusoids separated in phase by $90°$, we determine the sensor outputs when the surfaces are level to within a statistical limit of $\sim 100\,\mu$rad. By verifying that this measurement repeated to this accuracy several months later, we also tested the sensor's long term repeatability.

Now that we have a mirror surface which is normal to gravity ($\pm 50\,\mu$rad), we can make the bottom Raman beam vertical by forcing it to retro-reflect from this horizontal surface. However, the direction of the Raman beam is determined by mirrors which are rigidly attached to the optical table. The optical table floats on pressurized air legs which rest on the floor. The floor itself tilts with respect to gravity, and because of hysteresis in the passive displacement sensors which regulate the air pressure to the table legs, the table surface tilts even more than the floor. On a timescale of days, even with the experiment running, we have never observed the optical table to tilt by more than $350\,\mu$rad peak-to-peak. Every time we run, we reset the table to its level point. With the table leveled, once every several weeks we check the verticality of the bottom Raman beam. This alignment drifts by no more than $300\,\mu$rad. Between the alignment of mirrors with respect to the optical table and the drift of the table itself, we conservatively estimate that the Raman beams are never more than $\theta_g = 500\,\mu$rad misaligned from vertical. If the atom cloud is centered in the beam at the time of the early probe, by the time of the last interferometer pulse $\Delta t = 400$ ms later (assuming $T = 160$ ms and $N = 50$), the center of the atom cloud will have moved by $\Delta x_g = \frac{1}{2}g(\Delta t)^2\theta_g = \frac{1}{2}g(400\,\mathrm{ms})^2(500\,\mu\mathrm{rad}) = 1.1$ mm.

Of the three possible sources of motion across the beam: 1) Raman beams misaligned from vertical by θ_{g}, 2) launch off from vertical by θ_{L}, and 3) Raman beams not parallel by θ_{B}, the largest is $\Delta x_g \sim 1$ mm.

Longitudinal motion

The limit for the longitudinal distance along each Raman beam from the atoms to the beam's Gaussian focus is determined by how well we can position the collimating lenses. From Section 4.3 we estimate an accuracy of ±0.2 mm, which corresponds to an uncertainty in the focus position of 6.2 m.

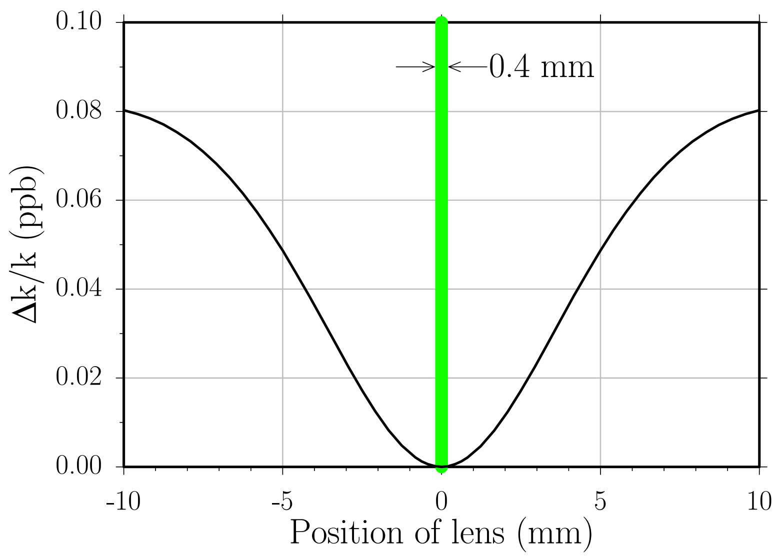


Figure 6.2: **Magnitude of the wavefront gradient for a $2w_0 = 2.0$ cm diameter Gaussian beam** in parts per billion (ppb) deviation from the gradient at the origin. The gradient is calculated at a distance of 2 m from the $f = 2$ m focal length collimating lens as a function of the longitudinal position of the lens, where 0 represents perfectly collimated. The width of the gray bar at the origin indicates the ±0.2 mm uncertainty in locating the collimation point of the lens.

As we move the collimating lens toward and away from the source, both the position and size of the Gaussian focus change. Figure 6.2 plots how the magnitude of the wavefront gradient at the atoms changes as we move the collimating lens. Within the estimated measurement accuracy of ±0.2 mm for the lens placement and a maximum off-axis displacement of 1 mm, the fractional wavefront gradient error is < 0.01 ppb from Figure 6.1 and < 0.001 ppb from Figure 6.2. Since the recoil frequency is proportional to k_{eff}^2, this corresponds to a measurement error of 0 ± 0.02 ppb.

Figure 6.3 provides experimental evidence that this effect is negligible. From the ±0.2 mm uncertainty in determining the collimation point of the lens and the measured sensitivity from this data, the error in $f_{\rm rec}$ due to both lenses could be at most 0.054 ± 0.16 ppb, consistent with no effect at the pbb level.

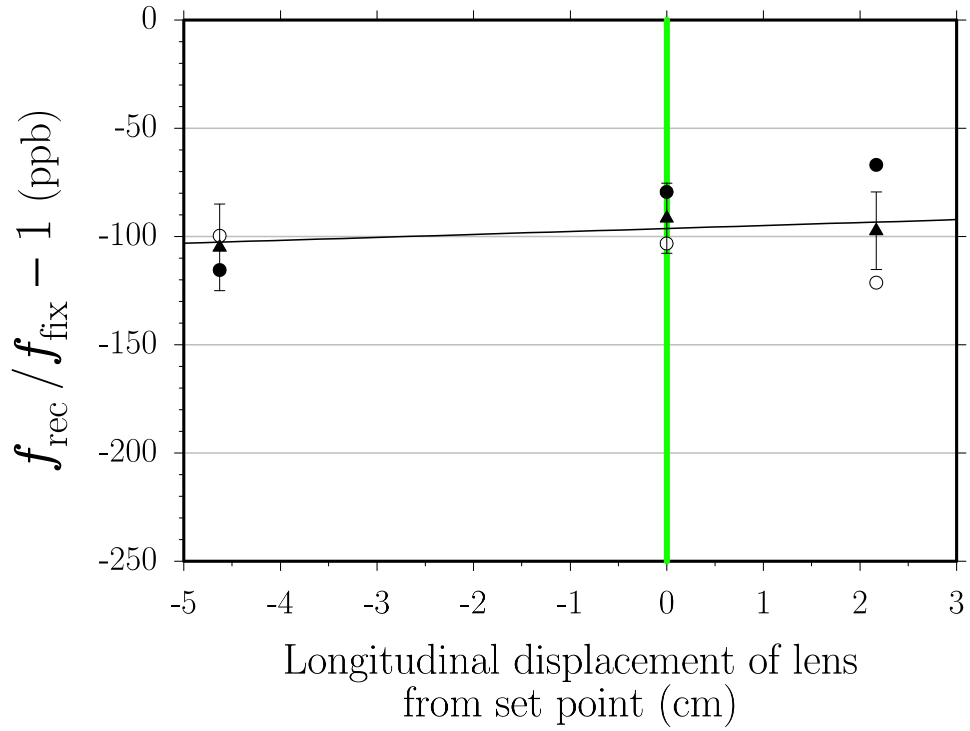


Figure 6.3: **Recoil frequency versus longitudinal displacement of the bottom collimating lens** from its set point. If one or both of the two $f = 2$ m lenses that collimate the Raman beams to a Gaussian diameter of ~ 2 cm are not set correctly, the laser wavefronts at the atoms will be curved and the local wavefront gradient which determines the size of the photon recoil will deviate from the expected value. Here we test the sensitivity to this effect by intentionally moving the lens away from the collimation point (see Section 4.3). The width of the vertical gray line represents the ± 0.2 mm uncertainty in determining this point. Fitting these data with a line gives intercept -96.2 ± 10.5 ppb and slope 1.37 ± 3.90 ppb/cm, consistent with no effect. For a detailed explanation of the graph symbols see [53].

6.1.2 Clipping

The sizes of our Raman beams are limited by the size of the smallest optic they encounter and by the 2.0 inch diameter of the hole in the endcaps of the magnetic shielding. The smallest Raman beam optics are the 2.0 inch polarizing beamsplitter cubes and the top elliptical mirror which also has an effective size of 2.0 inches. To test the effects of clipping, we installed in the top Raman beam a solenoid controlled blade ~ 21 cm before the top collimating lens. With current flowing through the solenoid, the blade was pushed into the beam to a repeatable distance of 8.3 mm from the beam center. With the solenoid off, a spring retracted the blade well outside of the beam. We took sensitive recoil data with $T = 135$ ms and 30 π-pulses, at each point alternating the solenoid on or off. The presence of the blade shifted the final measurement by $+18.3 \pm 20.2$ ppb, which is consistent with no effect. It is difficult to know exactly how to scale this result to predict the size of the effect from the clipping at 25.4 mm from the beam center. At worst, the effect is proportional to the electric field, which according to equation (6.1) scales as $\exp[-(r/w_0)^2]$, at a distance r from the beam axis. Therefore, extrapolating the size of the effect at $r_1 = 8.3$ mm to $r_2 = 25.4$ mm, we can reduce the size of the effect by

$$\begin{aligned} \exp\left[-(r_1/w_0)^2\right] \ / \ \exp\left[-(r_2/w_0)^2\right] &= \exp\left[-(r_1^2 - r_2^2)/w_0^2\right] \\ = \ \exp\left\{-\left[(8.3\,\mathrm{mm})^2 - (25.4\,\mathrm{mm})^2\right]/(9.6\,\mathrm{mm})^2\right\} \\ \simeq \ 520 \end{aligned}$$

and estimate an upper limit for the error from clipping the beams to be $+0.035 \pm 0.039$ ppb, consistent with no effect and small enough to neglect.

Another possible source of beam clipping are cables used during setup or the beam shielding tubes themselves. In general, the tubes prevent cables and other physical objects from accidentally blocking part of the beam. As an additional precaution, however, as part of the alignment process each time we take data, we use a video camera to image the scattering of both Raman beams off of a white card placed just above the top vacuum chamber window. In this image, we would immediately recognize if either of the beams was obstructed in some way.

6.1.3 Speckle

By speckle we describe the appearance of a laser beam after it has passed through or reflected off surfaces coated with small scatterers. Micro-defects in a dielectric coating, thin films, or dust particles could act as scattering centers with sizes ranging from one to several hundred optical wavelengths. How each scatterer affects the wavefronts of the lasers depends strongly on its size, shape, position relative to the beam, and distance from the atoms. Because of the complexity, it would be intractable to simulate this effect theoretically. Although we have not tested this effect with our experiment, we can refer to previous tests performed by colleagues on a similar apparatus measuring the local gravitational acceleration [54, 26]. Their experiment also uses light pulses driving two-photon transitions between the two cesium hyperfine ground states to construct an atom interferometer which can then be used to make a precision measurement. Like ours, their measurement depends directly on the quality of the laser wavefronts. In their test for the importance of speckle they inserted into their Raman beams a temporary glass plate onto which they placed varying amounts of baby powder. Not only was the final measurement independent to within $\sim$2 ppb of the amount of powder they added, but the contrast did not drop until the powder noticeably altered the appearance of the transmitted beam. Consequently, they estimated that the effect from any minute amounts of scatters on the optics would be less than 0.1 ppb. Unlike our experiment where we use dark-state transfer, their Raman lasers are tuned far away from resonance to avoid coupling to the short-lived excited state. Consequently, they do not have two independent beam paths. Their Raman beams copropagate through the vacuum chamber and then retroreflect off of a single mirror. As a result, the effect from the powder in the beam path may cancel to some degree, because their Raman beams encounter exactly the same optics and scattering centers. On the other hand, to the extent that the scattering process changes the spacing between optical wavefronts, the effect should *not* cancel. It is not clear whether such wavefront distortions will cancel in our experiment with some combination of the four fundamental interferometer geometries[2], although the discussion of clipping in Section 6.1.2 gives some indication of the size of the effect. At this point, we can say that we use high quality optics flat to better than $\lambda/10$ with a scratch-dig rating of 10-5 and that we inspect visually and clean all optics if

[2]Our previous work [9] states incorrectly that the effect of the distortion changes sign when the beam direction is reversed.

necessary every time we take data. In addition, because the distortion of a scatterer tends to propagate radially outward from the scatterer, the largest distortion occurs as one travels across the wavefront and not longitudinally in the beam direction.

6.1.4 Relative angle

According to equation (4.13), a relative angular misalignment between the two beams changes the measured recoil because it reduces the magnitude of k_{eff}. Our more precise method for minimizing this misalignment using pinholes is discussed in Section 4.4. To investigate the long term drift of this alignment we have observed the alignment signal continuously for over one week. After converting the tranmission signal through the bottom pinhole signal into a relative displacement and then into an a effective change of the recoil shift, we plot the results in Figure 6.4.

With the long horizontal time scale, it is difficult to see, but several times during this week, we realigned the top beam to the bottom beam just as we would do if we were running the experiment and taking data. Each time we realigned the beams, they immediately started drifting out of alignment. After studying their behavior shortly after each of the five times we reset the alignment, we propose the following model for approximating this behavior. Each time the beams were aligned, the alignment seemed to drift away at a rate of roughly -1 ppb/hour until it reached a level of approximately -1 ppb effective change in the recoil frequency measurement where it seemed to hold roughly constant. The mean measurement error from this whole data set is -1.02 ppb. In principle, we could correct all of our recoil data using this model. All of the data taken within an hour of each realignment would be corrected by a varying amount depending on when exactly it was taken. All of the remaining data would be corrected by simply adding 1.0 ppb. However, because of the inaccuracy involved in determining exactly how long after a particular realignment the data were taken, the uncertainty of the actual mean value it settles to, and because the vast majority of our data were taken over an hour after each realignment, we choose to correct all of our data by $+1.0$ ppb with an associated uncertainty of -0.3 to $+0.5$ ppb.

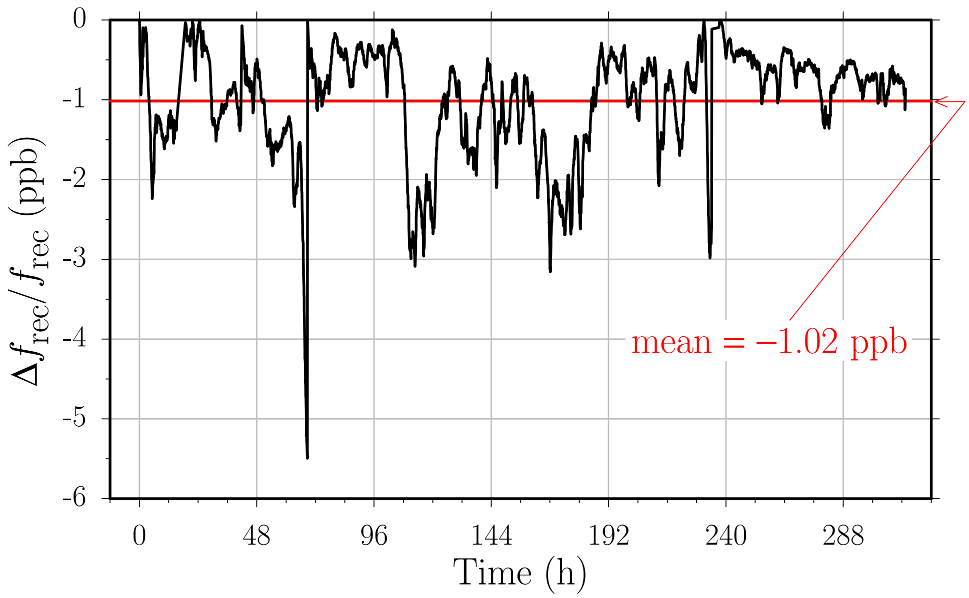


Figure 6.4: **Change in the recoil frequency $f_{\rm rec}$ due to a relative angular misalignment of the Raman beams.** Every 4 minutes we sample and record the transmission of the top Raman beam backward through the bottom pinhole (see Figure 3.11). We calculate the tranmission efficiency of a beam with Gaussian profile and diameter of $\sim 340\,\mu$m through a circular pinhole of roughly the same size as a function of the position displacement between the beam center and the pinhole. By inverting this function, we convert our transmission data into a relative displacement and with the 2 m focal length of the collimating lens into an angle $\theta_{\rm B}$. Using equation (4.13) we convert $\theta_{\rm B}$ into the relative change in $f_{\rm rec}$ which is the vertical axis. Five times during this over 13 day sample we realigned the top Raman beam as we would when running the experiment. The mean and standard deviation of this entire data set are -1.02 ppb and 0.67 ppb, respectively.

6.1.5 Polarization

In the lab frame of reference the two Raman beams are intended to have exactly the same perfectly circular polarization. If the beams are not perfectly $\hat{\sigma}_+$ polarized, they will couple other single-photon transitions besides ones that satisfy $\Delta m_F\!=\!+1$. The $m_F\!=\!0$ sublevels of the hyperfine ground states will no longer be coupled to only the $F\!=\!3', m_F\!=\!+1'$ excited state and the dark state, if one still exists, will be a more complex mix of the ground state magnetic sublevels. In general, if either one or both of the $m_F\!=\!0$ ground states are coupled to another excited state, there will be no purely dark state, because there is no general combination of the ground states which simultaneously cancel the coupling to both excited states. Any atoms that are transfered to an excited state via one of these addition couplings will then spontaneously emit a photon and lose any phase information they carried. If the only effect of additional couplings is to transfer of atoms out of the dark state, at worst it will slightly reduce the transfer efficiency and possibly decrease the interferometer contrast by increasing the incoherent background. On the other hand, a phase shift might result if the original dark state is altered by the presence of couplings to additional levels, either through an ac-stark shift or because the dark state now includes ground state magnetic field sensitive levels. In either case, because the shift results from a coupling to a magnetic field sensitive level, the shift will be linearly proportional to the magnetic bias field. By changing this bias field, we can rule out this effect (see Section 6.4).

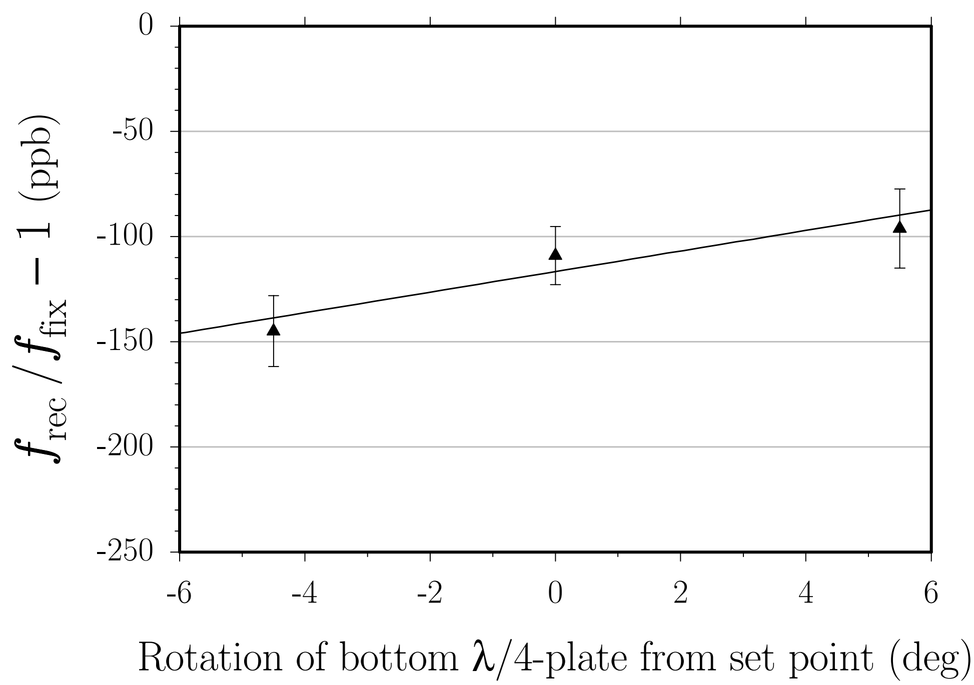


Figure 6.5: **Recoil frequency versus Raman beam polarization.** The bottom $\lambda/4$-plate was rotated about its axis to two values on either side of its set point where the contrast fell by more than 40%. A line with intercept -116.6 ± 9.3 ppb and slope $+4.9 \pm 2.5$ ppb/deg fits these data consisting of 117 total repetitions with $T = 135$ ms and 30 π-pulses. For a detailed explanation of the graph symbols see [53].

With the magnetic bias at its normal value of $\sim$72 mG, we have checked for effects from polarization impurity on two occasions by intentionally misaligning the bottom $\lambda/4$-plate. These results are shown in Figures 6.5 and 6.6. Because we have not

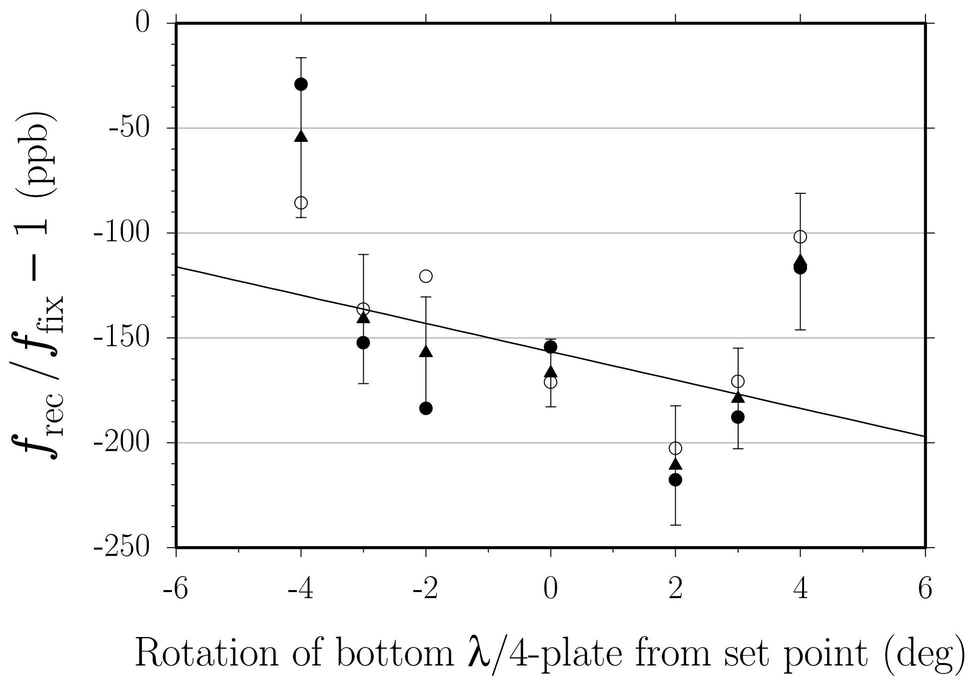


Figure 6.6: **Recoil frequency versus Raman beam polarization.** The bottom $\lambda/4$-plate was rotated about its axis to either side of its set point until the contrast dropped almost in half. A line with intercept -156.5 ± 9.6 ppb and slope -6.8 ± 4.0 ppb/deg fits these data consisting of 119 total repetitions with $T = 120$ ms and 30 π-pulses. For a detailed explanation of the graph symbols see [53].

proposed a specific model for how the polarization impurity will shift the final phase, it is impossible to say whether any dependence will be linear or quadratic in its lowest order. As a worst case estimate, though, we will look for a linear effect. Averaging the two slopes from Figures 6.5 and 6.6 and combining their single-standard deviation uncertainties in quadrature results in a slope of $+1.6 \pm 2.1$ ppb/deg, consistent with no effect.

Since the set point in Figures 6.5 and 6.6 is the best alignment point for the waveplate, we must now estimate how accurately we can determine this point. As discussed in Section 3.2.7, the beam polarization is set by passing each beam through a polarizing beamsplitter (PBS) cube and then through a zero-order $\lambda/4$-plate. To determine the set point for the waveplates we block the top beam and above the vacuum chamber place a mirror whose surface we set to be normal to the upward-going bottom beam. We insert a non-polarizing beamsplitter (BS) in the bottom beam before the collimating lens. This BS allows us to detect with a photodiode the light returning from the retro-reflection back through the $\lambda/4$-plate and the PBS. If the bottom $\lambda/4$-plate is set correctly, the returning light will be orthogonally linearly polarized with respect to the incoming beam after passing twice through the waveplate. The PBS will reflect this orthogonally polarized light. Thus, the light returning to the photodetector will be minimized when the waveplate is optimally oriented.

To first order these waveplates can be modeled as a perfect crystal whose optic axis lies in the plane of the plate and is oriented at an angle θ from the incoming linear polarization. Because of the crystal's birefringence, light passing through the waveplate polarized along the optic axis ($\theta = 0$) will experience a propagation phase retardation or advance of ϕ with respect to light polarized orthogonal to the optic axis ($\theta = \pi/2$). In the lab frame with a basis defined by the two possible orthogonal polarizations $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ of the incoming laser beam, this idealized waveplate transforms the input polarization $\binom{x_0}{y_0}$ according to

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos^2\theta + \sin^2\theta e^{-i\phi} & \cos\theta\sin\theta(1 - e^{-i\phi}) \\ \cos\theta\sin\theta(1 - e^{-i\phi}) & \cos^2\theta e^{-i\phi} + \sin^2\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{6.9}$$

A perfect $\lambda/4$-plate would thus have $\phi = \pi/2$ at the operating wavelength and be optimally positioned when $\theta = \pi/4$. With an input polarization of $\binom{1}{0}$ equation (6.9) would become

$$\begin{pmatrix} \frac{1}{2}(1-i) & \frac{1}{2}(1+i) \\ \frac{1}{2}(1+i) & \frac{1}{2}(1-i) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1-i \\ 1+i \end{pmatrix} = \frac{1}{2}(1-i)\begin{pmatrix} 1 \\ i \end{pmatrix} = \hat{\sigma}_+ \tag{6.10}$$

To minimize the light returning to the photodetector we turn the waveplate about the beam axis so as to vary θ and slightly tilt the plate with respect to the beam to vary ϕ. For small angles, changing the angle of incidence slightly varies the effective thickness of the waveplate and thus the net propagation phase difference ϕ.

Once the bottom $\lambda/4$-plate is set, we remove the retroreflecting mirror and observe the upward-going bottom beam after it passes through the top $\lambda/4$-plate and the top PBS. By minimizing this transmitted signal, we optimize the two parameters θ and

ϕ for the top waveplate. Although this method of aligning the top waveplate using the light polarization produced by the bottom waveplate may not result in the best circular polarization of the top beam, it does insure that the polarization of the top beam is as identical as possible to the polarization of the bottom beam. Well matched polarizations between the top and bottom beams produce the best dark state. By repeating this alignment procedure and comparing the resulting values of θ, we estimate that we can set each waveplate to with in $\pm 0.2°$. After aligning both waveplates, the best ratio of minimum over maximum transmission through the PBS we can achieve is better than 2×10^{-3}. Using equation (6.9), this extinction ratio is equivalent to an angular misalignment of $\theta \sim 0.97°$. By independently measuring the isolation of the PBS cube to be $< 5 \times 10^{-4}$, we absolve the polarizer and conclude that the light polarization is limited by the quality of the waveplate. From the repeatability of our alignment procedure, we guess that the alignment limit is better than 2×10^{-3}. However, without further tests, a safe alignment uncertainty would be $\pm 0.5°$.

Another possible systematic alignment error results from the possible birefringence of the vacuum chamber windows. To check for window birefringence we aligned the bottom waveplate twice, once with the retro-reflecting mirror above the chamber and once with the mirror below the chamber, and compared the two alignment positions. The two alignments differ by approximately $\Delta\theta = 0.685 \pm 0.027°$ and $\delta\phi = 0.15 \pm 0.03°$, indicating some birefringence in one or both of the two vacuum chamber windows. Because the atoms are located between the two vacuum windows, the polarization impurity at the atoms will be largest if the top window produces all of the birefringence. If the bottom window is solely to blame, then when the bottom waveplate is set using a retro-reflecting mirror above the chamber, the waveplate will compensate for the birefringence of the bottom window, so that the atoms experience nearly circular polarized light. Because the birefringence of the windows most likely results from atmospheric pressure and the metal vacuum flange stressing the glass, we assume that whatever birefringence the windows have is equally distributed between the two windows. For this reason, we halve the value for $\Delta\theta$ and $\Delta\phi$. However, because both Raman beams contribute to the polarization impurity, we must then double the effect. From equation (6.9) it can be shown that the polarization impurity of the electric amplitude scales linearly with small deviations from the correct value of θ or 4ϕ. If we assume that there is some systematic effect due to polarization impurity and that the effect is proportional to the intensity of light in the wrong polarization, we can add the contributions of $\Delta\theta$ and $4\Delta\phi$ in quadrature to arrive at an effective angular offset $\Delta\theta' = (\Delta\theta)^2 + (4\Delta\phi)^2 = \pm 1.67(37)^\circ$. If we assume that the 2×10^{-3} extinction limit is not due to polarization impurity, we can reduce this offset to $\Delta\theta' = \pm 0.96(37)^\circ$. Combining this offset with the mean slope from Figures 6.5 and 6.6, we assign a systematic uncertainty of ± 2.0 ppb from polarization impurity.

A final source of polarization impurity is a misalignment between the magnetic bias field and the beam direction. The magnetic field generated by the bias and bias trim coils serves to define a quantization axis and to remove the degeneracy of the magnetic sublevels. If the direction of this field makes an angle ϵ with the beam direction, the polarization that the atoms experience will be $\hat{\sigma}_+ + \epsilon\hat{\Pi}$. We have no easy way of measuring or adjusting the direction of the magnetic bias field. However, as mentioned above, any effect from polarization impurity should depend linearly on the size of the magnetic bias field, so by varying the bias field (see Section 6.4) we can, in fact, test for this possible misalignment.

6.2 Frequencies

In the same way that the position of the laser wavefronts provide a reference to compare with the atom's spatial evolution, the laser frequencies are the local oscillator to which we compare the time evolution of the atom's internal states. To accurately define their wavelength, the two Raman beams must be frequency locked to cesium. The difference of their frequencies which serves as the local oscillator for the interferometer must be phase locked to a stable microwave reference. Furthermore, because the two Raman beams counterpropagate, for every pulse we must change the frequencies of each beam so that their difference compensates for the two-photon Doppler shift which changes with the atom's velocity as it recoils and accelerates due to gravity. Between the first and second and the third and fourth $\pi/2$-pulse, when the atom freely evolves without light in a superposition state, it is most important that the frequencies change in a phase stable and repeatable manner. The layout and control of Raman laser frequencies is discussed in Section 3.3. In this section we are concerned with how this process may go wrong and if it does, how it affects the final measurement.

6.2.1 Lock to cesium

We can use the cold atoms from the atomic fountain to test the long term accuracy of our lock to cesium. By programming the Raman beams to flash on a single velocity selecting π-pulse, we can accurately determine the time in the trajectory when the atoms reach the top of their trajectory and momentarily come to rest. The pre-selection π-pulse uses dark-state adiabatic transfer to drive the atoms from the $F=4$ to $F=3$ ground state with a full width half maximum (FWHM) linewidth of $\sim$38 kHz (see Figure 3.32), much smaller than the single-photon linewidth of 4.6 MHz. This linewidth combined in quadrature with the effective frequency width of the atoms' velocity distribution gives a total width of 76 kHz (see Figure 3.32). If we set the two-photon difference frequency to resonance (i.e. with Doppler shift compensation set to zero) and then vary the time of the pulse until the transfer efficiency is maximal, this time will correspond to the top of the trajectory when the atoms have zero mean velocity. Assuming perfect statistics from 50 points, the uncertainty of this determination will be approximately $\pm[(76\,\text{kHz})/\sqrt{8\ln 2}]\lambda_{\text{eff}}/\sqrt{50} = \pm 2.0$ mm/s. We then block the $F=3$ Raman beam, flash on the $F=4$ light, and scan the Raman Cs-lock offset frequency across the single-photon resonance. We fit these data by a Lorentzian lineshape and determine the center to about $\pm$68 kHz. By repeating this measurement after months and even several years, we conclude that the lock to cesium[3] is long-term accurate to within $\pm$100 kHz.

If both Raman lasers are detuned by Δ from the cesium transition as in equation (2.68), the size of the recoil will change according to

$$\Delta k_{\text{eff}}/k_{\text{eff}} = +2\Delta/f_{\text{eff}} = +2\Delta\ \lambda_{\text{eff}}/c = +2.98\,\text{ppb/MHz} \tag{6.11}$$

[3]Note that in order to measure the recoil frequency we do not need to know what the exact transition frequency is, only that it is locked to cesium.

Since the recoil shift is proportional to k_{eff}^2, the recoil frequency changes as

$$\Delta f_{\text{rec}}/f_{\text{rec}} = 2\Delta k_{\text{eff}}/k_{\text{eff}} = +4\Delta\ \lambda_{\text{eff}}/c = +5.96\,\text{ppb/MHz} \tag{6.12}$$

Thus, if the both laser frequencies have detuning $\Delta = 0 \pm 100$ kHz, the final measurement should be corrected by 0 ∓ 0.60 ppb.

6.2.2 Difference frequency

The laser difference frequency is used as the local oscillator which tracks the time-dependent part of the atomic evolution. As discussed in Section 5.1, for each atomic launch we slightly detune the two-photon difference frequency of the final two $\pi/2$-pulses from resonance by an amount $\delta = f_s$, allowing us to scan the phase $\phi = 2\pi f_s T$ across the fringes. If $f_s = 0$ does not correspond to resonance, the numerical fit will incorrectly determine the phase zero point. However, any detuning present for one interferometer will produce exactly the same phase shift for its conjugate and will thereby cancel with the up/down difference. Two additional effects which might also depend on two-photon detunings are a sloping background (Section 6.8.1) and dispersive features (Section 6.5).

To verify that there is no effect from two-photon detunings, we vary δ for the final two $\pi/2$-pulses as far as approximately ± 20 kHz. The results are shown in Figures 6.7 and 6.8. As a lowest order approximation to some δ-dependence, we fit each of these data sets with a line. Then, to arrive at a systematic error from this detuning, we estimate the largest possible error in determining the resonance frequency. It is due to our uncertainty in the local gravitational acceleration[4] g. Our neighbors located $\sim$10 m away within 1 m of the same elevation using another precision atom interferometry apparatus have measured $g = 9.799\,33\cdots$ to better than 10 ppb [14, 26]. However, what truly limits our knowledge of the acceleration due to gravity is an uncertainty in the verticality of our Raman beams. In Section 6.1.1 we estimated that the Raman beams

[4] Even if our value for f_{rec} were incorrect by 200 ppb, after 50 π-pulses, this would be a frequency error of only $f_{\text{rec}}(N+1)(\Delta f_{\text{rec}}/f_{\text{rec}}) = 15\,\text{kHz}(50+1)200\,\text{ppb} = 0.15$ Hz.

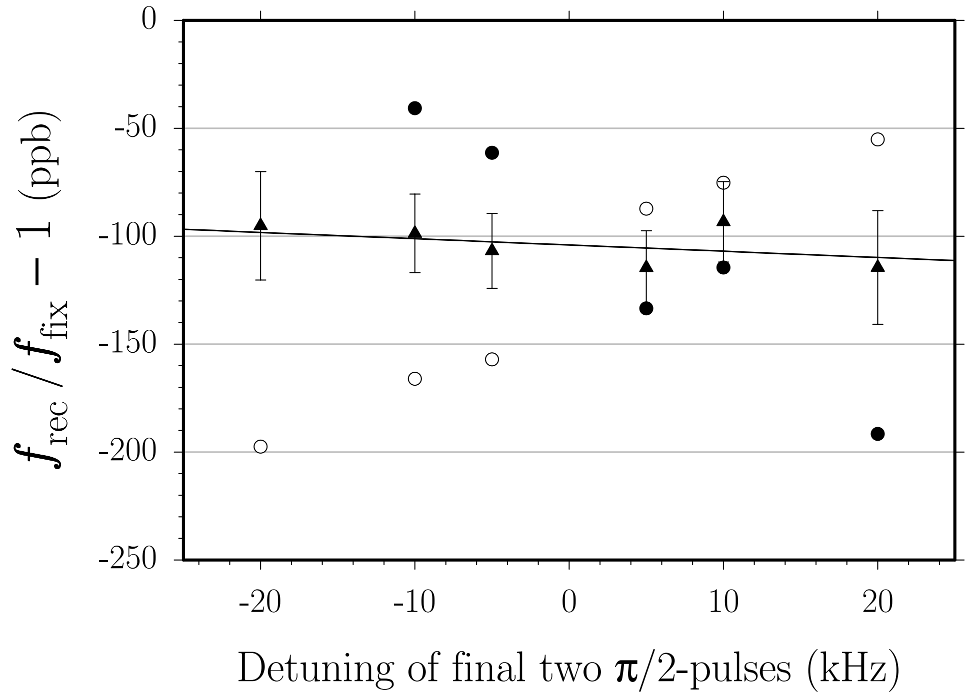


Figure 6.7: **Recoil frequency versus two-photon detuning** of the last two $\pi/2$-pulses. A line with intercept -104.0 ± 8.0 ppb and slope -0.29 ± 0.71 ppb/kHz fits these data consisting of 86 total repetitions with $T = 135$ ms and 30 π-pulses. For a detailed explanation of the graph symbols see [53].

are misaligned from vertical by at most $\theta_{\rm g} = 0.5$ mrad. To lowest order this misalignment changes the value of g by $\Delta g/g = -\theta^2/2 = -(0.5\,\text{mrad})^2/2 = -125$ ppb. If the lasers are on resonance at the velocity-selecting π-pulse, they will be detuned by at most $g/\lambda_{\rm eff}(T + T' + T)(\Delta g/g) = (22\,\text{MHz/s})(160 + 30 + 160\,\text{ms})(125\,\text{ppb}) = 0.96$ Hz at the last interferometer pulse. This maximum possible detuning times the mean slope from the linear fits in Figures 6.7 and 6.8 of $+0.13 \pm 0.41$ ppb/kHz gives a negligible error of $(12 \pm 39) \times 10^{-5}$ ppb in determining $f_{\rm rec}$.

We also varied the value of $\bar{g}$ used to calculate the atomic trajectory by $\pm 0.3\%$. If this value does not match the actual gravitational acceleration g, the calculated resonance frequencies for all of the pulses after the first velocity-selecting π-pulse will be wrong by $0.03(21.9\,\text{MHz/s})(0.13\,\text{s}) = 8.54$ kHz on average for interferometers with $T = 120$ ms, 30 π-pulses, and $(\bar{g}/g - 1) = 0.3\%$. From all of the data taken when we varied $\bar{g}$, the weighted mean variation in $f_{\rm rec}$ is -121.5 ± 30.9 ppb/(%g). Since our

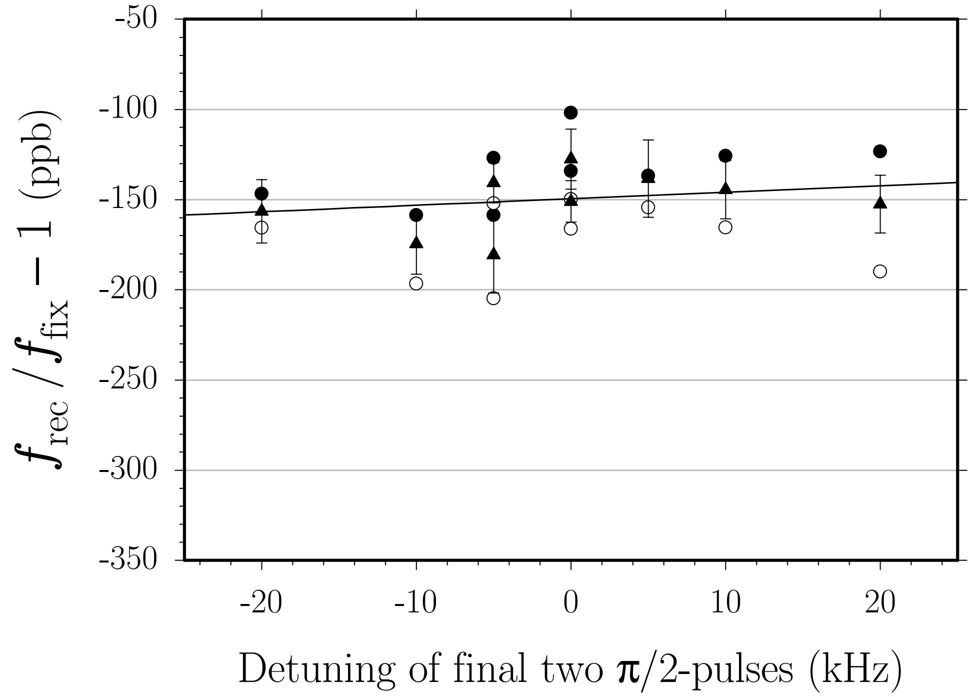


Figure 6.8: **Recoil frequency versus two-photon detuning** of the last two $\pi/2$-pulses. A line with intercept -145.6 ± 5.1 ppb and slope $+0.35 \pm 0.51$ ppb/kHz fits these data consisting of 323 total repetitions with $T = 135$ ms and 31 π-pulses. For a detailed explanation of the graph symbols see [53].

value for g can be wrong by no more than $-125\,\text{ppb} = -1.25 \times 10^{-5}\%$, this slope is equivalent to a negligible final measurement error of $+0.0015 \pm 0.0004$ ppb.

6.2.3 Difference frequency switching

As described in Section 3.3.1 the phase of the rf output of a direct digital synthesizer (DDS) is compared with the phase of the mixed down beatnote between the two Raman beams. The Raman phase lock loop (PLL) detects any change in this phase difference and controls a 9 GHz voltage-controlled oscillator (VCO) to compensate.

With the synthesizer running at some constant output frequency f_1, although the short-term stability will be limited by the spectral purity of the synthesizer's output, the noise floor of the microwave beatnote, and the noise above ~ 1 Hz inherent in all electronics, the long-term stability of the phase difference between the two

Raman beams will be excellent. In fact, ultimately it will be limited only by the long-term stability of the LORAN C time standard. It is another story entirely when the synthesizer changes its output frequency to f_2. Here we depend on the synthesizer to change frequency in a repeatably phase continuous and predictable way (see Section 4.1). We also require that all of the electronics that handle the changing frequencies have no frequency dependent propagation delay. Consider how such dispersive behavior affects the final interferometer phase Φ. According to Section 2.1.1, the phase sensitivity of our four $\pi/2$-pulse interferometers is given by equation (2.19): $\Phi = \phi_1 - \phi_2 + \phi_3 - \phi_4$, where ϕ_i is the phase at the ith $\pi/2$-pulse. At each $\pi/2$-pulse occuring at time t_i, the DDS is set to output some frequency f_i. Since the synthesizer changes to the next frequency at times t_{12}, t_{23}, and t_{34}, (see Figure 2.1) the final interferometer phase will be

$$\Phi = f_1(t_{12} - t_1) + f_2(t_2 - t_{12}) + f_3(t_{34} - t_3) + f_4(t_4 - t_{34}) \tag{6.13}$$

Consider first a constant signal propagation delay t_d. Because of this delay, equation (6.13) must be modified by

$$\Delta\Phi = t_d[(f_1 - f_2) + (f_3 - f_4)] \tag{6.14}$$

which is proportional to the delay t_d and the two frequency differences. For our interferometers, these frequency differences have opposite sign when N is even and the same sign with N is odd. They are all equal[5] to $\Delta f_{12} = (-1)^{(N+1)}\Delta f_{34} = \pm g/\lambda_{\text{eff}}T$, where the sign depends on the direction of the $\mathbf{k}_{\text{eff}}$. For instance, for the normal interferometers $\boxed{1}$ and $\boxed{2}$ with $N = 0$, equation (6.14) becomes

$$\Delta\Phi = t_d[(g/\lambda_{\text{eff}}T) - (g/\lambda_{\text{eff}}T)] = 0 \tag{6.15}$$

[5]This is not exactly true. Although the *desired* frequency shifts are equal, the synthesizer cannot always output these exact frequencies. As discussed in Section 4.1, the frequencies it may output are limited to integer multiples of $f_{\text{clk}}/2^{32} = 0.232$ Hz. Thus the frequency differences will be identical to within 0.23 Hz. If the delay t_d is as long as 2 μs , an upper limit for the phase error from this inexact cancelation is $2\,\mu\text{s}\,(0.23\,\text{Hz}) = 2.9\mu$rad, which corresponds to less than 0.01 ppb and can be ignored.

If, however, the delay depends on the frequencies before and after the change, the phase error $\Delta\Phi$ will not in general vanish. Because f_1 and f_2 are the same for the up and down interferometers, there will be some cancelation when the final phase shifts are subtracted. However, f_3 and f_4 are never the same and in fact can vary arbitrarily depending on where in the atom's trajectory the interferometers start. We have extensively tested and verified that the DDS changes frequency in a predictable manner (see Section 4.1). The remaining sections of the Raman PLL which see changing frequencies have not been tested. These components include the rf part of the PLL electronics up to and including the phase detector chip, the high-frequency beatnote photodiode, amplifier, and microwave mixer before the Raman PLL box. The components designed to operate at microwave frequencies can probably be absolved because the sizes of the frequency changes we care about (~ 4 MHz) are much less then their maximum operating bandwidth. For the lower frequency rf components, we can estimate the effect by modeling them as simple filters. A first-order low pass filter with corner frequency f_c has transmission phase given by

$$\phi = -\tan^{-1}\left(\frac{f}{f_\mathrm{c}}\right) \;=\; -\left[\frac{f}{f_\mathrm{c}} - \frac{1}{3}\left(\frac{f}{f_\mathrm{c}}\right)^3 + \frac{1}{5}\left(\frac{f}{f_\mathrm{c}}\right)^5 + \cdots\right] \tag{6.16}$$

From equation (2.19), we have the change $\Delta\phi$ in the final interferometer phase difference for a four $\pi/2$-pulse interferometer

$$\Delta\phi = \phi(f_1) - \phi(f_2) + \phi(f_3) - \phi(f_4) \tag{6.17}$$

where f_i is the laser difference frequency for the ith $\pi/2$-pulse. The frequency f_2 of the second $\pi/2$-pulse differs from f_1 because of gravity: $\Delta f_{12} = f_2 - f_1 = -kgT$, and similarly for last $\pi/2$-pulses: $\Delta f_{34} = +kgT$, for the normal interferometers. Since $\Delta f_{34} = -\Delta f_{12} = \Delta f$, the linear term in equation (6.16) does not contribute to $\Delta\phi$. Inserting the next order term proportional to $(f/f_\mathrm{c})^3$ into equation (6.17) gives

$$\Delta\phi \;=\; \frac{1}{3f_\mathrm{c}^3}\left[f_1^3 - (f_1 - \Delta f)^3 + f_3^3 - (f_3 + \Delta f)^3\right]$$

$$= \frac{(f_1^2 + f_3^2)\Delta f}{3f_c^3} + \mathrm{O}(\Delta f^2) \tag{6.18}$$

Since f_1 is the same for the up and the down interferometer, the up/down difference leaves only

$$\Delta\phi \simeq \frac{f_3^2 \Delta f}{3f_c^3} \tag{6.19}$$

to lowest order in Δf. For a representative case with $T = 120$ ms and 30 π-pulses, $\Delta f \simeq 2.6$ MHz and the frequencies for the third $\pi/2$-pulse are $f_3(\boxed{1}) \simeq 13.1$ MHz and $f_3(\boxed{2}) \simeq 12.2$ MHz for interferometers $\boxed{1}$ and $\boxed{2}$, respectively. Thus, after the up/down difference we have

$$\Delta\phi \simeq \frac{[(13.1\,\mathrm{MHz})^2 - (12.2\,\mathrm{MHz})^2](2.6\,\mathrm{MHz})}{3f_c^3} = \frac{19.7\,\mathrm{MHz}^3}{f_c^3} \tag{6.20}$$

The closest filter element is the $f_c = 80$ MHz bandwidth of the phase detector chip in the Raman phaselock loop electronics (see Appendix B), giving a phase error of $\Delta\phi = 19.7/80^3 = 39\,\mu$rad, equivalent to an error in f_{rec} of 0.055 ppb. Since this worst case estimate is much smaller than the 0.4 ppb limit from the DDS discussed in Section 4.1, we neglect it.

For an experimental demonstration that the DDS no longer causes >100 ppb errors, consult Figure 6.18 where we varied the time in the atoms' trajectory when the interferometer sequence occurs. The mean atomic velocity and therefore the difference frequency required to correct for the Doppler shift changes with this interferometer start time. Thus, for each of these points, the DDS had to output different frequencies. The data in this figure are consistent with no variation, and thus we know that the error due to DDS switching can be know larger than the peak-to-peak variation of 13.8 ± 24.9 ppb.

We also compare data taken with N and $N+1$ π-pulses. On six separate occasions for a total of 186 repetitions we took data with both 31 and 30 π-pulses. We subtract the results for $N = 31$ from the results for $N = 30$ and compute a weighted averaging all of these differences, given $f_{\mathrm{rec}}(N=31) - f_{\mathrm{rec}}(N=30) = -6.8 \pm 12.9$ ppb, consistent with no effect.

6.2.4 Gravity chirp

Originally, the Raman beam frequency difference was held constant during the $\pi/2$-pulses. Because gravity continues to change the atomic velocities throughout the $T_{\pi/2} = 250\ \mu$s long pulse, the lasers are resonant at only one time and then shift away from resonance at a rate of $\pm g/\lambda_{\rm eff} = \mp 22$ MHz/s, where the sign depends on the direction of $\mathbf{k}_{\rm eff}$. To minimize the detuning from resonance, the frequencies were set to be resonant at the center of the pulse. Thus, the phase shift at the beginning and end of the $\pi/2$-pulses is

$$\phi(T_{\pi/2}) = \mp\frac{1}{2}2\pi\frac{g}{\lambda_{\rm eff}}\left(\frac{T_{\pi/2}}{2}\right)^2 = \mp 2.15\,\text{rad} \tag{6.21}$$

changing at a rate of

$$\frac{\partial\phi}{\partial t}(T_{\pi/2}) = \mp 2\pi\frac{g}{\lambda_{\rm eff}}\left(\frac{T_{\pi/2}}{2}\right) = \mp 17.2\times 10^3\,\text{rad/s} \tag{6.22}$$

If the falling edge of the first $\pi/2$-pulse is not the exact time reversal of the rising edge of the second $\pi/2$-pulse such that the time when the dark-state projection occurs differs by as little as 58 ns, there will be a $\phi_1 - \phi_2 = 1$ mrad phase error in the final interferometer phase difference, which corresponds to a 1.4 ppb error for interferometers with $T = 120$ ms and 30 π-pulses. As mentioned in Section 4.5 this asymmetry could occur if the beam intensities ($F\!=\!3$ up, $F\!=\!4$ down, $F\!=\!3$ down, $F\!=\!4$ up) were not all equal.

To greatly reduce the size of this effect, we now linearly sweep the two-photon difference frequency to cancel the Doppler shift due to gravity (see Section 3.6.3). To test the importance of chirping the frequencies to stay resonant, we varied the magnitude and even the sign of the chirp rate. From the slope of the linear fit to the data in Figure 6.9, not chirping produces an error of -22.1 ± 5.6 ppb. It should be noted that when these data were taken, we were not using a common switch AOM, so the sensitivity to the frequency chirp rate might have been greater. But, even though we now use the common switch AOM to turn the two Raman beams on

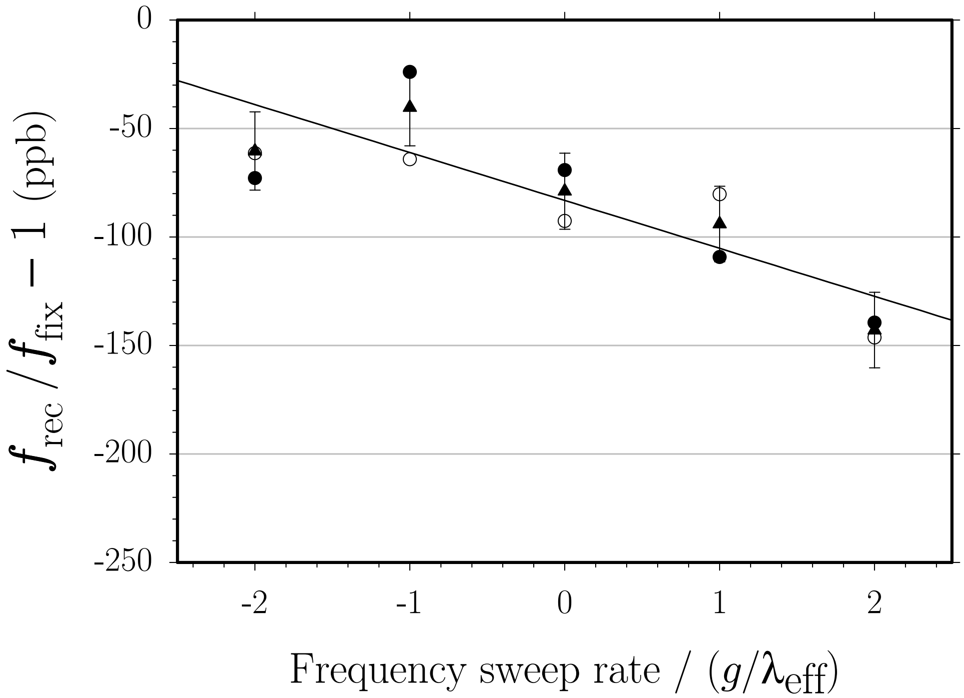


Figure 6.9: **Recoil frequency versus two-photon frequency sweep rate** during each of the four $\pi/2$-pulses. A line with intercept -83.1 ± 7.9 ppb and slope -22.1 ± 5.6 ppb/($g/\lambda_{\rm eff}$) fits these data consisting of 100 total repetitions with $T = 135$ ms and 31 π-pulses. For a detailed explanation of the graph symbols see [53].

and off simultaneously, to minimize any potential error, we still chirp the difference frequency.

6.2.5 Gravity gradient

The downward acceleration of an atom of mass m due to the attractive gravitational force of the Earth can be written as

$$g = \frac{F_g}{m} = G\frac{M_{\rm E}}{r^2} \simeq G\frac{M_{\rm E}}{R_{\rm E}^2} \tag{6.23}$$

where G is Newton's gravitational constant and $M_{\rm E}$ is the mass of the earth. The distance $r \simeq R_{\rm E} = 6.38 \times 10^6$ m is the displacement between the atom and the Earth's center of mass. To the lowest two orders, the fractional change in g over a distance

Δr in the lab

$$\begin{aligned}
\frac{\Delta g}{g}(\Delta r) &= \frac{1}{g(R_{\mathrm{E}})}\left(\left.\frac{\partial g}{\partial r}\right|_{R_{\mathrm{E}}}(\Delta r) + \frac{1}{2}\left.\frac{\partial^2 g}{\partial r^2}\right|_{R_{\mathrm{E}}}(\Delta r)^2\right) \\
&= \frac{R_{\mathrm{E}}^2}{GM_{\mathrm{E}}}\left(-\frac{2GM_{\mathrm{E}}}{R_{\mathrm{E}}^3}\Delta r + \frac{1}{2}\frac{6GM_{\mathrm{E}}}{R_{\mathrm{E}}^4}(\Delta r)^2\right) \\
&= -2\left(\frac{\Delta r}{R_{\mathrm{E}}}\right) + 3\left(\frac{\Delta r}{R_{\mathrm{E}}}\right)^2
\end{aligned} \tag{6.24}$$

Over the maximum separation of $\Delta z(\text{up}-\text{down}) \sim 12$ cm from equation (6.8) between the up and down interferometers for $T = 160$ and $N = 50$, the value of g will differ by at most $\Delta g/g = -2(12\,\text{cm})/(6380\,\text{km}) = -38$ ppb. From equation (2.21) for interferometer $\boxed{1}$ and equation (2.33) for interferometer $\boxed{2}$, this difference in the value for g will shift the final interferometer phase of the up/down difference by at most $-2(2\pi g/\lambda_{\text{eff}})(\Delta g/g)(T + T')T = -4\pi(22\,\text{MHz/s})(-38\,\text{ppb})(0.16\,\text{s} + 0.03\,\text{s})(0.16\,\text{s}) =$ 0.31 rad. Without the inverted interferometers which reverse the roles of the up and down interferometers, this shift would cause an error of over 200 ppb. The inverted interferometers have the exact same spatial separation between the final $\pi/2$-pulses, so the constant and linear gradient term of equation (6.24) both cancel leaving only the second-order term. This term does not exactly cancel because both inverted interferometers are $\Delta z(\text{normal} - \text{inverted}) = v_{\text{r}}(2T + T') = 1.7$ mm lower than the normal interferometers, so for $T = 160$ ms with 50 π-pulses, the final phase difference from all four interferometers will be at most $(0.34\,\text{rad}(1.7\,\text{mm})/(6380\,\text{km}) =$ 9.1×10^{-11} rad, which is negligible for this experiment.

6.2.6 Bad frequencies

When the DDS is set to 12 631 770 Hz, the Raman beam difference frequency will be set to exactly the ground state hyperfine splitting. Although this is a significant frequency for the cesium atom, it is nothing special to the control electronics. Conversely, there are frequencies which are significant to the control electronics but not especially noteworthy from the cesium atom's point of view. For example, when the beam direction and the atom's velocity are such that the Doppler shift is −2 631 770 Hz from resonance, the DDS will be set to output 10.0 MHz, which is a frequency common in the lab. The LORAN C time reference is distributed by a 10 MHz sinusoidal signal. All of the Raman beam synthesizers are locked to copies of this signal. The switchyard and common switch AOM frequencies are also small integer multiples of this frequency. Consequently, many of the rf sources exhibit small amounts of amplitude modulation at 10 MHz. Via the AOMs, this modulation makes it onto the light and is then converted back to an electronic signal at the high-frequency beatnote. It will still be present on the mixed down version of the beatnote which the phaselock compares to the output of the DDS. The phaselock electronics will interpret this modulation as phase error oscillating at the frequency difference between the DDS output and the nearest integer multiple of 10 MHz. For most DDS output frequencies, this amplitude converted to phase modulation will occur at frequencies large enough to produce only a small amount of random phase noise. However, when the DDS output frequency is set near 10 MHz, this modulation can be slow and repeatable enough to consistently change the final interferometer phase difference. As discussed in Section 4.6, we have greatly reduced this problem by installing crystal filters in many of the rf signal lines, and have thus significantly reduced any 10 MHz sidebands. As an additional precaution, we also avoid atomic trajectories where the center frequencies of the pulses need to be tuned near one of these sensitive frequencies.

6.2.7 Computer arithmetic

A final issue dealing vaguely with the laser frequencies concerns the numerical calculation of the atomic trajectories. A single computer program (see Appendix C.1) written in BASIC complied by version 4.50 of *Microsoft's* QuickBasic for DOS programs all of the synthesizers and runs the entire experiment. It calculates the atomic trajectories from which it computes the center frequencies for each of the pulses. Even if one is careful to minimize error prone numerical arithmetic, the accuracy required to calculate the frequencies easily exceeds that of single precision floating point arithmetic. Consequently, double precision is used for all variables involved in the frequency calculation. As a check, we calculated the trajectory and center frequencies for several different interferometer configurations using another computer running a completely different program written in C instead of BASIC. With the same inputs, the two computers agree to better than 1 mHz, which is sufficient for our purposes.

6.3 Electric fields

6.3.1 dc-Stark effect

By far the dominant source of external electric fields near the atoms are the lasers themselves. However, if there is a static electric field present, there will be a dc-stark shift described by

$$\Delta f_{\mathrm{DC}} = \alpha_E E^2 \tag{6.25}$$

where $\alpha_E = -1.00 \times 10^{-5}\,\mathrm{Hz}/(\mathrm{V/m})^2$ for cesium [55]. Since this shift is approximately the same for both hyperfine ground states, it changes only the final interferometer phase whenever the field varies over the average separation $\Delta z = v_{\mathrm{r}}(T + T')$ between the two interferometer paths. Just as we did in equation (6.33) for the spatial variations of the magnetic field, substituting the lowest order term of equation (6.25) which is not zero gives

$$\begin{aligned} \Delta f_{\mathrm{DC}} &\simeq \alpha_E (2 E_0 \frac{\partial E}{\partial z} \Delta z) \\ &\simeq 2 \alpha_E E_0^2 \frac{\Delta z}{d} \end{aligned} \tag{6.26}$$

where d is the approximate size of the object carrying a static charge. The electric field in units of V/m required to introduce a 1 ppb measurement error is therefore

$$\begin{aligned} E_0^2 &\simeq \frac{f_{\mathrm{rec}}(N+1)}{10^9} \frac{d}{2\alpha_E v_{\mathrm{r}}(T+T')} \\ &= \frac{1}{2 \times 10^9 \alpha_E} \frac{(N+1)}{(T+T')} \frac{d}{\lambda_{\mathrm{eff}}} \\ &= 5 \times 10^{-5} \frac{(N+1)}{(T+T')} \frac{d}{\lambda_{\mathrm{eff}}} \end{aligned} \tag{6.27}$$

For typical sensitive interferometers with $T = 120$ with 30 π-pulses (implying $T' \simeq$ 20 ms), the electric field E_0 would have to be ~ 160 V/m for the $d \sim 1$ m vacuum chamber and ~ 35 V/m for the $d \sim 2$ inch diameter aluminum cylinder holding the magnetic bias coil windings. The aluminum cylinder is the physical object closest to the atoms that might carry a charge[6]. However, because it is conducting and electrically shorted to the vacuum chamber which is grounded, it is difficult to imagine how such a static field would develop or persist.

There is also a differential dc-stark effect which changes the ground state hyperfine splitting f_{34} by

$$\frac{\Delta f_{34}}{f_{34}} = [-2.5 \times 10^{-20}\,(\mathrm{V/m})^{-2}]\,E^2 \tag{6.28}$$

which for cesium gives $\Delta f_{\mathrm{hfs}} = [-2.3 \times 10^{-10}\,\mathrm{Hz/(V/m)^2}]\,E^2$ [47]. The field required to change the recoil frequency by 1 ppb is

$$\begin{aligned} E^2 &= \frac{2 f_{\mathrm{rec}} T}{10^9} 4.3 \times 10^9\,\frac{(\mathrm{V/m})^2}{\mathrm{Hz}} \\ &= 8.6 f_{\mathrm{rec}} T\,\frac{(\mathrm{V/m})^2}{\mathrm{Hz}} \end{aligned} \tag{6.29}$$

For $T = 120$ ms, this field is ~ 120 V/m, which is roughly the same limit we estimated for the common mode dc-stark effect. In summary, shifts from static electric fields at the ppb level are easily avoidable by using grounded metal vacuum chambers.

6.3.2 AC-Stark effect

For ac-stark shifts, we are concerned with shifts of the form

$$\Delta f_{\mathrm{AC}} = \frac{\Omega^2}{4\Delta} \tag{6.30}$$

where Ω is the Rabi frequency defined in equations (2.66) and (2.67) from some oscillating field source such as a laser and Δ is its detuning from resonance. During the interferometers when the atoms are sensitive to frequency and phase shifts, all

[6]Because of the grounded vacuum chamber, free ions are not likely to be present in significant quantities.

of the lasers except the tracer and Raman lasers are mechanically shuttered (see Figure 3.5). The slowing beam shutter blocks the trap and the repumping light to the slowing beam path. The trap shutter blocks the remaining trap and repumping light to the MOT. The Zeeman pumping beam shutter blocks this beam and its accompanying repumping light. The probe shutter blocks the on-resonance probe and clearing beams plus the far-detuned Doppler-free Raman beams. Besides the light from the interferometer lasers, the only remaining light comes from the lights in the room. Since the overhead lights are always off when we take data, the only other sources of background light that might reach the atoms are the small quantities from LED and LCD front panel displays, oscilloscopes, computer monitors, *etc.* Since they are in general relatively weak and spectrally broad, the light shift from these sources can be safely neglected.

Tracer laser

The tracer laser must be turned on before each interferometer pulse to give the tracer phaselock feedback loop enough time to settle completely. To be safe, unless it is already on, we turn the tracer on 1.8 ms before each $\pi/2$-pulse. Thus, between the first and second and third and fourth $\pi/2$-pulses when the atoms are freely evolving in a superposition state and the light is supposed to be off, the tracer is on for 1.8 ms. If during this time, it were to perturb the ground state cesium levels, a change in the final measurement value might result. The tracer is roughly 100 times less intense than the Raman beams and it is detuned from resonance by no less than $\Delta_{\text{tracer}} = 748$ GHz. It could therefore shift the $F\!=\!3$ ground state by $\Delta f_3 = \frac{1}{4}\frac{1}{100}(\Omega = 2.5\,\text{MHz})^2/(748\,\text{GHz}) = 21$ mHz. It would shift the $F\!=\!4$ state by the same amount except for the detuning difference due to the hyperfine splitting f_{34}. The net change in the difference frequency is thus $(\Delta f_{34}) = (f_{34}/\Delta_{\text{tracer}})\Delta f_3 = (9.19\,\text{GHz}/748\,\text{GHz})(21\,\text{mHz}) = 0.26$ mHz. During each of the four 250 μs long $\pi/2$-pulses, this shift from the tracer might cause a $2\pi(0.26\,\text{mHz})(250\,\mu\text{s}) = 0.40\,\mu\text{rad}$ phase shift. Even if the phase shift from each pulse accumulated and did not cancel, it would change the measurement of the recoil frequency negligibly. If this ac-stark shift were present for the whole time T and it did not cancel between interferometers,

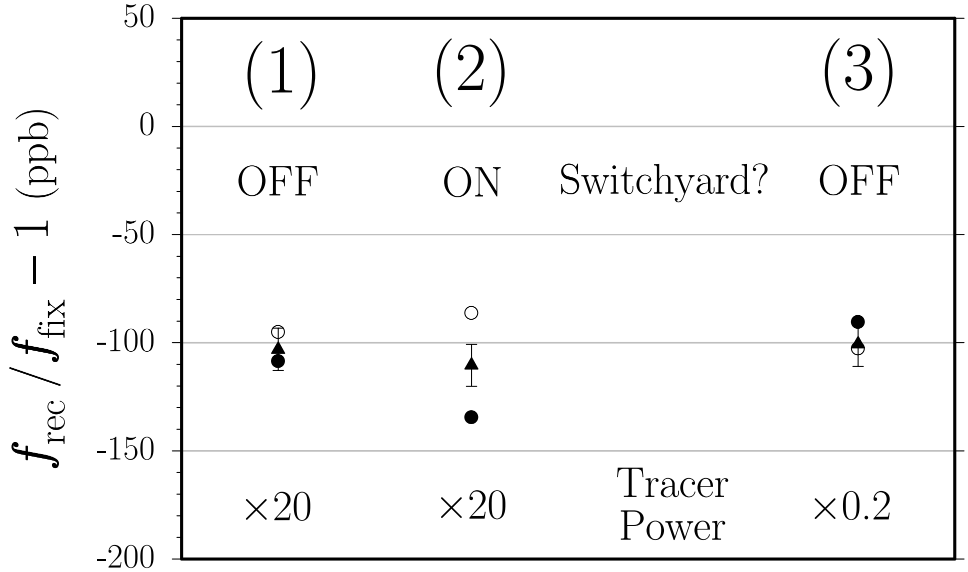


Figure 6.10: **AC-stark effect from the tracer laser.** Here we compare 215 repetitions of $T = 120$ ms with 40 π-pulses for three different conditions for the tracer laser. In cases (1) and (2), we have $\sim$20 times *more* tracer light than normal, compared with case (3) where we used $\sim$5 times *less* tracer power than normal. In cases (1) and (3), the tracer is switched off between the pulses as normal. For case (2), however, the tracer is kept on the whole time. Thus, the total effect of light shifts from the tracer should be approximately of 6700 times bigger for case (2) than for case (3). Since these points differ by 10 ± 25 ppb, the ac-stark shift from the tracer must be less than $(1.5 \pm 3.7) \times 10^{-3}$ ppb. For a detailed explanation of the graph symbols see [53].

it would change the recoil measurement using 30 π-pulses by $\Delta f_{34}/(f_{\text{rec}}(N+1)) =$ 0.55 ppb. Since the tracer is on for only 1.8 ms out of the time T, for $T = 120$ ms, for example, this error should be further reduced to $1.8/120(0.55\,\text{ppb}) = 8.3 \times 10^{-3}$ ppb. In addition, it should cancel with the up and down interferometer difference.

To verify that the tracer does not affect the measurement during the free-evolution times, we take recoil data with the tracer off between the $\pi/2$-pulses as normal and with the tracer on during the whole time. For $T = 120$ ms, this should increase any ac-stark shift effect by a factor of $\sim 120/1.8 = 67$. To further increase the size of the effect, we temporarily replaced the tracer laser diode with fiber coupled light from a third Ti-sapphire laser from another lab. After the top and bottom fibers we had approximately 20 times more light for use as a tracer laser. To make the comparison as fair as possible, we tuned the Ti-sapphire output wavelength to within 0.01 nm of the tracer laser diode's output wavelength. Figure 6.10 shows the final results for three experimental conditions: 1) 20 times more tracer power than normal but off between the pulses as normal, 2) 20 times normal power and on the whole time, and 3) 0.2 times normal power and switched off between pulses. The ac-stark shift effect should be $\sim 100(67) = 6700$ times larger for condition 2) than for condition 3), so since the difference between the resulting recoil values for these two conditions is 10 ± 25 ppb, an experimental upper limit for the ac-stark shift from the tracer is $(1.5 \pm 3.7) \times 10^{-3}$ ppb. Note that this test also tends to rule out any effect from glitches on the tracer phase lock loop (PLL) error signal due to Raman light leaking onto the tracer photodiode.

Raman laser

The interferometer lasers are shut off with an isolation of better than 6×10^{-10} (see Table 3.1). Assuming a full intensity Rabi frequency of 2.5 MHz (see Appendix A.1.2) and a representative detuning of $\Delta \sim 1$ MHz, according to equation (6.30) the ac stark shift from one of the Raman beams would change the ground state hyperfine splitting by $\Delta f_{34} \simeq 4$ mHz. If this shift were present during the whole time T when the atoms are supposed to be evolving in the dark, the measured value of f_{rec} would change by 4 ppb for interferometers with $T = 120$ ms and 30 π-pulses. However, if some light from the other Raman beam also leaked out of the fiber, it would shift the other hyperfine ground state by approximately the same amount and thus tend to greatly reduce the change in Δf_{34}. In addition, if the light leakage is the same for the up and down interferometers, the phase shifts will also be the same and tend to subtract out.

Unlike the tracer, we cannot as easily turn the Raman beams on and off or change their intensities by orders of magnitude without adversely affecting the transfer efficiency. We can, however, significantly change the one-photon detuning Δ and still drive two-photon transitions. Figures 6.11 and 6.12 show the results from two separate times when we varied the one-photon detuning of both beams. Combining the slopes from these two data sets we have an upper limit of $+0.16 \pm 1.02$ ppb/MHz from the effect of single-photon detuning errors. To estimate a final measurement error, we must also estimate an upper limit for Δ. This is determined by the long-term accuracy of our lock to cesium described in Section 3.3.2. In Section 6.2.1 we explain

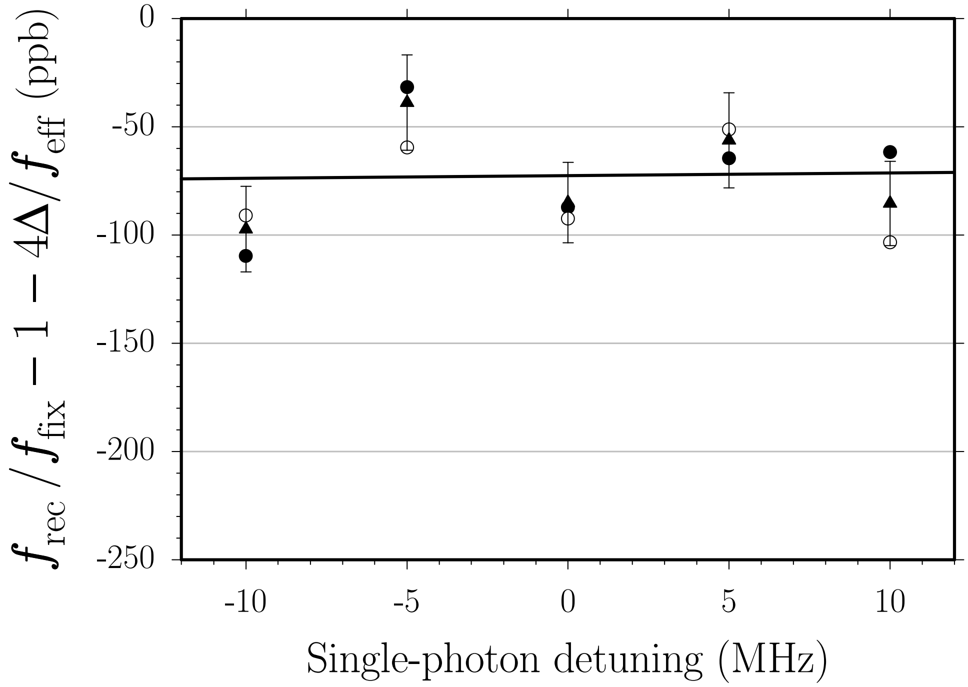


Figure 6.11: **Recoil frequency versus the single-photon detuning of Raman lasers.** The vertical axis is corrected according to equation (6.12) for the +5.98 ppb/MHz change of the momentum recoil due to the change the laser frequency. The resulting data set consisting of 53 repetitions with $T = 120$ ms and 30 π-pulses fits a line with intercept -108.9 ± 9.3 ppb and slope $+0.20 \pm 1.31$ ppb/MHz, consistent with no effect. For a detailed explanation of the graph symbols see [53].

how we arrive at the number of $|\Delta| < 100$ kHz. An ac-stark shift from the Raman beams should be inversely proportional to the single-photon detuning from resonance. Using the combined slopes from Figures 6.11 and 6.12 and the measured ± 100 kHz accuracy of the lock to cesium, we have an upper limit for the total ac-stark shift due to the Raman beams of 0.016 ± 0.102 ppb.

6.4 Magnetic fields

The $m_F = 0$ magnetic sublevels are not completely insensitive to magnetic fields. The term proportional to x^2 in equation (3.19) gives the quadratic Zeeman shift of the

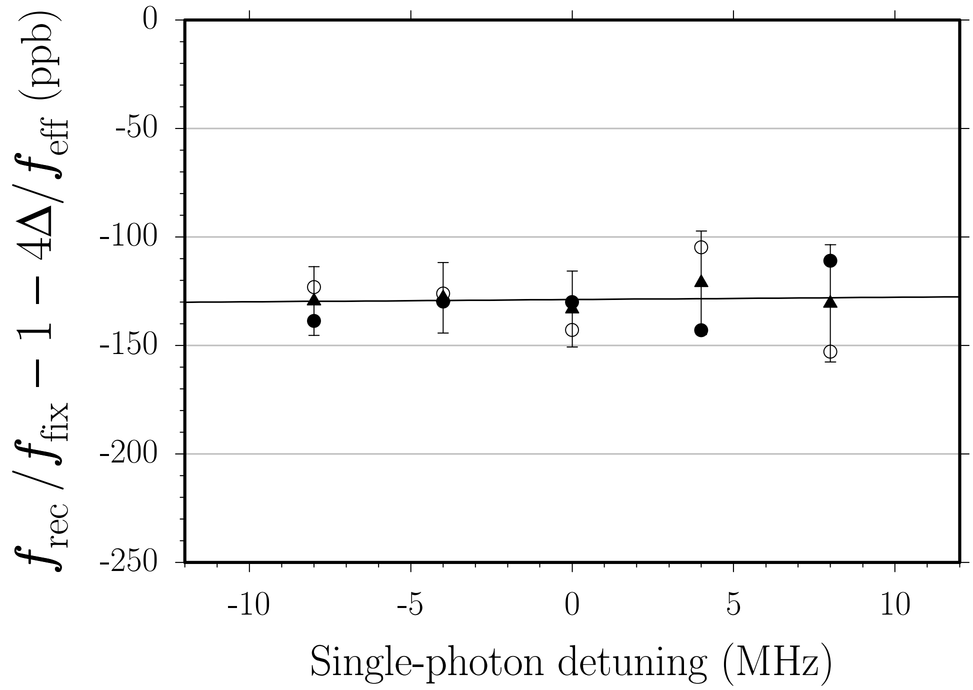


Figure 6.12: **Recoil frequency versus the single-photon detuning of Raman lasers.** The vertical axis is corrected according to equation (6.12) for the +5.98 ppb/MHz change of the momentum recoil due to the change the laser frequency. The resulting data set consisting of 165 repetitions with $T = 135$ ms and 30 π-pulses fits a line with intercept -123.9 ± 9.1 ppb and slope $+0.10 \pm 1.64$ ppb/MHz, consistent with no effect. For a detailed explanation of the graph symbols see [53].

ground state hyperfine splitting

$$\begin{aligned} \Delta f_B &\simeq \frac{(g_J + g_I)^2 \mu_{\mathrm{B}}^2}{2 f_{34}} B^2 \\ &= (4.274\,5 \times 10^{-4}\,\mathrm{Hz/mG^2})\, B^2 \end{aligned} \tag{6.31}$$

To calculate exactly how this level shift would change the final interferometer phase, we would have to integrate this expression over the path traced out by the interferometer, accumulating a phase shift both when the atoms are freely evolving in a superposition state and also when they are in the same internal state but spatially displaced. This calculation would be quite complex, because the atom's position depends strongly on when the interferometer pulses occur during the trajectory. So, instead of attempting to calculate it exactly and correct the final measurement, we designed the magnetic bias field (see Section 3.5) so that the effect for our atomic trajectories would be small. In this section we predict and show experimentally that the error due to magnetic field inhomogeneities is on the order of a few ppb.

The spatial variation of the magnetic field can be written

$$B(z) = B_0 + \frac{\partial B}{\partial z}\Delta z + \cdots \tag{6.32}$$

Thus, B^2 becomes

$$\begin{aligned} B^2(z) &= \left(B_0 + \frac{\partial B}{\partial z}\Delta z + \cdots\right)^2 \\ &\simeq B_0^2 + 2B_0\frac{\partial B}{\partial z}\Delta z \end{aligned} \tag{6.33}$$

A static magnetic field uniform in space introduces an identical phase shift for both conjugate interferometers. Thus, the phase shift from the B_0^2 term in equation (6.33) cancels in the up/down difference expression. The remaining term in equation (6.33) will not in general cancel because the up and down interferometers trace out different paths in space. To estimate how much phase shift this will cause, we refer to the spatial dependence of the magnetic field shown in Figure 3.19. The largest absolute slope in this graph is approximately $\partial B/\partial z \leq 0.15$ mG/cm. For an extreme case where $T = 160$ ms with 50 π-pulses, from equation (6.8) the up an down interferometers are separated by $\Delta z \simeq 12$ cm. If the linear magnetic field gradient existed over this entire range, the difference in the resonance frequency shift between the up and down interferometers would change by

$$\begin{aligned} \Delta f_B &= \Delta f(\text{up}) - \Delta f(\text{down}) \\ &\simeq \alpha_B 2B_0\frac{\partial B}{\partial z}\Delta z \qquad (6.34) \\ &= (4.27\times10^{-4}\,\text{Hz/mG}^2)2(71.6\,\text{mG})(0.15\,\text{mG/cm})(13\,\text{cm}) \\ &= 0.12\,\text{Hz} \qquad (6.35) \end{aligned}$$

which is equivalent to a measurement error of

$$\Delta f_B/(2(N+1)f_{\rm rec}) = 0.12\,\text{Hz}/(2(51)15\,\text{kHz}) = 72\,\text{ppb} \tag{6.36}$$

Fortunately, we also have inverted interferometers which reverse the role of up and down and thereby the sign of this effect. The normal/inverted values for Δf_B cancel to the extent that the normal and inverted interferometers trace out the exact same paths. For the extreme case with $T = 160$ ms and 50 π-pulses, the inverted interferometers occur $\Delta z(\text{normal} - \text{inverted}) = v_{\rm r}(2T + T') = 1.7$ mm lower than the normal interferometers, so the remaining error might be as large as $72\,\text{ppb}(1.7\,\text{mm})/(12\,\text{cm}) = 1.0$ ppb.

To experimentally verify this cancelation, we take data with different magnetic bias field values up to $B_0 = 2600$ mG. From Figure 6.13 the opposite effects for the normal ($\bullet$) and inverted ($\circ$) interferometer pairs is clear. Figure 6.14 shows the same data but with the vertical axis amplified to show cancelation from the normal/inverted averaging (6). For these data we altered the bias field by changing the current to the bias coil, but we did not change the currents to the two bias trim coils. Because the bias trim coils are set for the normal operating field level of $B_0 = 71.6$ mG, as the bias field increases, the trim coils no longer correctly compensate to minimize the field variation within the magnetic shielding. As a result, the variation within the shield $\sim\partial B/\partial z$ also scales with the bias field value B_0. We would therefore expect a quadratic dependence on B_0.

In principle, both the linear and quadratic terms represent a measurement error for non-zero bias field values. To reduce the systematic uncertainty due to the linear term, we took more data (see Figure 6.15) zooming in on the region from 0 to 600 mG. Combining the slope from these data with the linear term from Figure 6.13 gives a final slope of -0.014 ± 0.028 ppb/mG. Extrapolating to $B_0 = 0$ from our normal operating field level of $B_0 = 71.6$ mG gives errors of -1.0 ± 2.0 ppb and $+0.150 \pm 0.098$ ppb from the linear and quadratic terms, respectively. Since we expect a non-zero effect from the magnetic field, in addition to folding these error bars into the final systematic uncertainty, we also modify the final value of $f_{\rm rec}$ by $+1.0 - 0.15 = +0.85$ ppb to

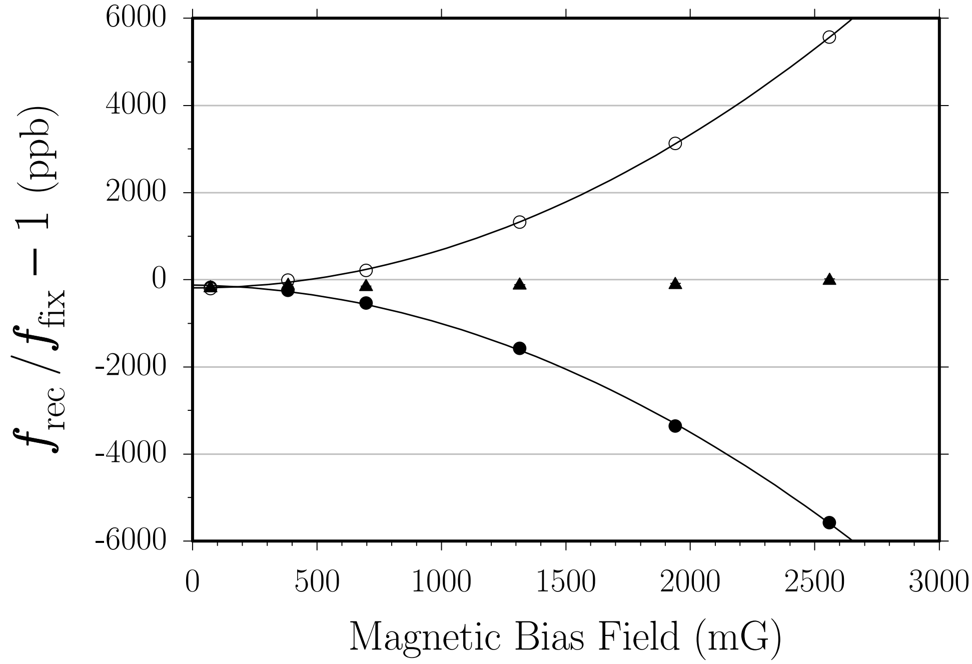


Figure 6.13: **Recoil frequency versus magnetic bias field.** Both the normal interferometer difference (solid circles •) and the inverted interferometer difference (hollow circles ∘), with all of the recoils reversed, show a strong dependence on the magnetic bias field. However, the effect is almost exactly opposite in sign, so for the normal/inverted mean (solid triangles), the effect effectively cancels. The fit results using a second order polynomial of form $\Delta f_{\mathrm{rec}}/f_{\mathrm{rec}} = a_0 + a_1 B_0 + a_2 B_0^2$ are shown below. Note that for the mean of the normal and inverted interferometers, the coefficients a_1 and a_2 are consistent with zero.

	a_0 (ppb)	a_1 (ppb/mG)	a_2 (ppb/mG^2)
$\boxed{1} - \boxed{2}$ • normal •	−115(59)	−0.081(107)	$-8.05(37) \times 10^{-4}$
$\boxed{3} - \boxed{4}$ ∘ inverted ∘	−182(33)	−0.003(060)	$+8.80(21) \times 10^{-4}$
$\frac{1}{2}\left(\boxed{1} - \boxed{2} + \boxed{3} - \boxed{4}\right)$ 6 normal/inverted 6	−154(30)	−0.022(054)	$+0.29(19) \times 10^{-4}$

correct for this measured systematic error.

As discussed in Section 6.1.5, it is also possible that some of the $m_F \neq 0$ magnetic field sensitive sublevels could be included in the dark state. According to equation (3.21), with a magnetic bias field level of 72 mG, the frequency separation between the $F=3, m_F=1$ and $F=4, m_F=1$ states differs from the $F=3, m_F=0 \to F=4, m_F=0$

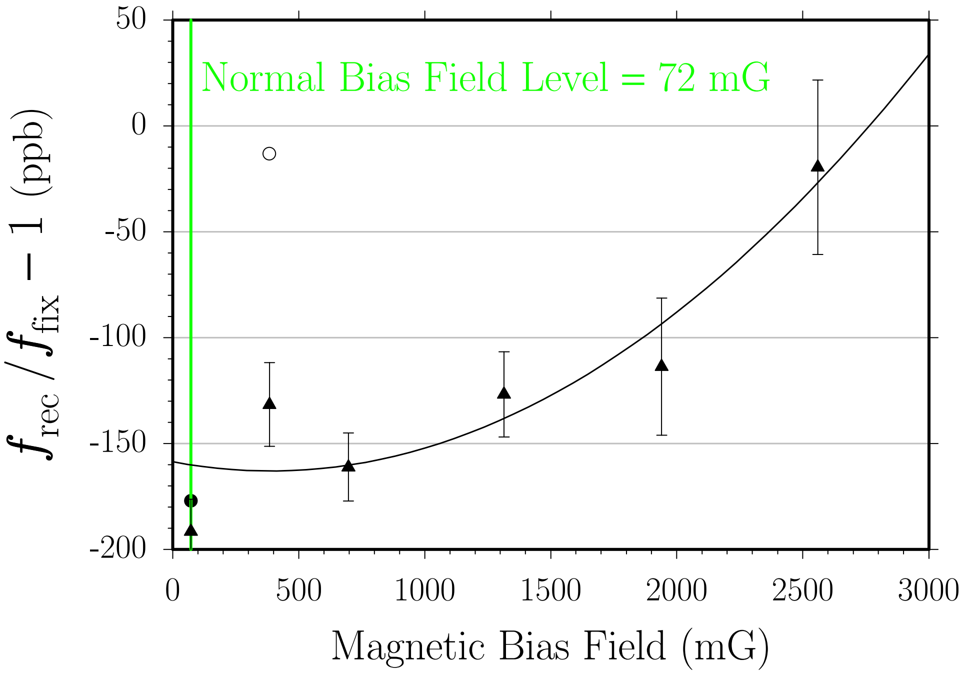


Figure 6.14: **Recoil frequency versus magnetic bias field.** This figure shows the same data as in Figure 6.14 but with the vertical scale magnified by a factor of 48. The normal operating magnetic bias field level of 72 mG is shown in gray.

transition frequency f_{34} by $\Delta f_{34} = 50$ kHz, which results in a final interferometer phase different by $\Delta\phi = (2\pi)2\Delta f_{34}T$. This difference in phase is over 1000 rad even for interferometers as short as $T = 5$ ms. Because of the spatial variation of the magnetic field, this phase shift can easily vary by 2π or more over the finite size of the atom cloud. These different phase shifts from different regions of the cold atom cloud will tend to counteract each other and wash out the effect.

6.5 Dispersion

Dispersion affects the recoil measurement by delaying (or advancing) the position of the Raman beam wavefronts as the laser frequencies change. Dispersive media the beams encounter include the optical glass the beams passes through, the background gases in the vacuum chamber, and the cold cesium cloud.

A dispersive effect from the optical glass (primarily BK7 from *Schott*) that the

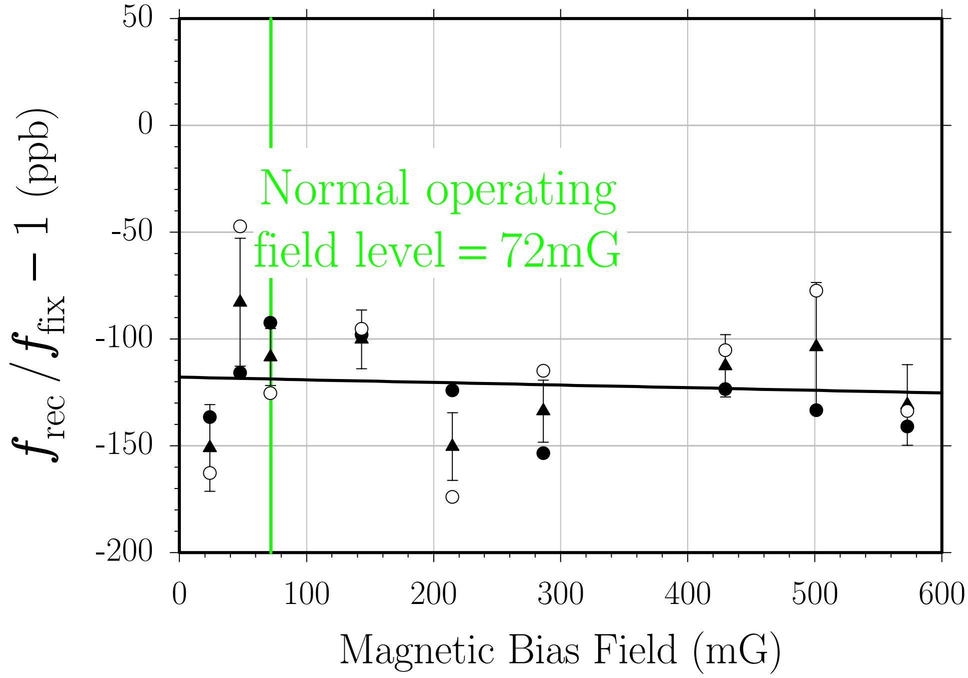


Figure 6.15: **Recoil frequency versus magnetic bias field.** The normal operating magnetic bias field level of 72 mG is shown in gray. A line with intercept -117.9 ± 9.7 ppb and slope -0.012 ± 0.033 ppb/mG fits these data consisting of 359 total repetitions with $T = 135$ ms and 30 π-pulses. For a detailed explanation of the graph symbols see [53].

beams pass through can be neglected for two reasons. First, any disturbance of the wavefront spacing is local to the particular optic and does not persist throughout the space where the interferometers take place. Second, the glass material has no sharp resonance features near the cesium wavelength. Similarly, non-cesium atoms in the vacuum chamber will be so far out of resonance with the Raman lasers that their dispersive effect over the relatively small range of frequencies we must tune to can be neglected. We therefore need consider only the cesium atoms found in the vacuum chamber.

When a laser detuned by Δ from a single atomic resonance with linewidth Γ enters a gas of that species, its wavenumber k changes by

$$f(\Delta) = \frac{\Delta k}{k} \simeq \frac{n\sigma}{2}\frac{\lambda}{2\pi}\left(\frac{-2\Delta}{\Gamma}\right)\frac{1}{1 + (2\Delta/\Gamma)^2} \tag{6.37}$$

assuming the transition is not greatly saturated. The absorption cross-section

$$\sigma = \sigma_{12} = \frac{3\lambda^2}{2\pi}\alpha(1 \to 2) \tag{6.38}$$

depends on the particular transition from state $|1\rangle$ to $|2\rangle$ being addressed. The parameter $\alpha(1 \to 2)$ discussed in Appendix A.2 is a constant representing the angular part of the transition strength. Our lasers are tuned to the D1 $F = 3'$ excited state of cesium, so we first calculate the fractional change in the wavenumber due to the $F = 3 \to F = 3'$ and $F = 4 \to F = 3'$ transitions with $\lambda = 894.60$ nm and linewidth $\Gamma = 2\pi(4.58\,\text{MHz})$.

The values of $\alpha(1 \to 2) = \alpha(\text{D1})$ for the cesium D1 line in Table A.3 assume the atom is randomly oriented with respect to the light polarization, which is reasonable for the atoms launched from the MOT. The resulting scattering cross-sections are also shown in Table A.3.

To calculate the total wavenumber change we must convolve expression (6.37) with the atomic velocity distribution, because an atom moving with non-zero velocity in the beam direction will be Doppler shifted from resonance. The velocity distribution expressed in terms of the Doppler shift $\Delta = v/\lambda$ is given by

$$g(\Delta) = \frac{\lambda}{\sigma_v\sqrt{2\pi}} \exp\left(-\frac{\lambda^2\Delta^2}{2\sigma_v^2}\right) \tag{6.39}$$

where σ_v is the rms velocity of the atoms. The convolution of equations (6.37) and (6.39) is

$$\begin{aligned} F_{12}(\Delta) &= \int_{-\infty}^{+\infty} f(\Delta - \Delta')g(\Delta')\mathrm{d}\Delta' \\ &= -A_{12}\int_{-\infty}^{+\infty} \left(\frac{\Delta - \Delta'}{\Gamma}\right) \frac{1}{1 + 2\left(\Delta/\Gamma\right)^2} \exp\left(-\frac{\lambda^2\Delta'^2}{2\sigma_v^2}\right) \mathrm{d}\Delta' \end{aligned} \tag{6.40}$$

where

$$A_{12} = \frac{n\sigma_{12}\lambda}{4\pi}\frac{\lambda}{\sigma_v\sqrt{2\pi}} \tag{6.41}$$

is a constant which depends on the initial and final state.

6.5.1 Room temperature background gas

Since the background cesium atoms in the vacuum chamber are in thermal equilibrium with the room, they must have a temperature of $T = 273\text{K} + 20\deg C = 293\text{K}$ and consequently have an rms velocity of $\sigma_v = \sqrt{2k_{\text{B}}T/m} = 191$ m/s. This rms velocity corresponds to an effective Doppler full-width half-maximum (FWHM) of $\Delta_v = \sigma_v/\lambda\sqrt{8\ln 2} = 503$ MHz for a laser tuned near the D1 transition at $\lambda = 894.60$ nm.

We use the background signal from the atom detection to estimate the density of the room temperature cesium atoms in the vacuum chamber. This background signal is $\sim$1100 times smaller than the peak fluorescence signal from the atoms when the probe laser is flashed on at 520 ms (late probe) after the launch when the cold atoms are passing through the detection region on their way downward. Because the room temperature atoms have such a large effective Doppler width, only a fraction of roughly $\Gamma(\lambda/\sigma_v) = 5.23\,\text{MHz}/(503\,\text{MHz}) = 0.010$ are detected by the probe pulse. Another factor of $\frac{1}{2}$ should also be included, because on average only half of the atoms will be in the $F\!=\!4$ state addressed by the probe laser. This implies that the density of the room temperature atoms is $\sim 1100(0.010)/2 = 5.7$ times smaller than the density of the cold atom cloud at the time of the late probe[7].

To estimate the cold atom density at the late probe we assume a simple atomic spatial distribution of a sphere whose radius is the $\sqrt{\pi/2}$ times the rms radius $r(t)$ of the atomic cloud. Inside the sphere all of the atoms distributed with a uniform density of

$$n(t) = \frac{3}{4\pi}\frac{N_0}{\left(r(t)\sqrt{\pi/2}\right)^3} = \frac{3}{\pi^{5/2}\sqrt{2}}\frac{N_0}{r^3(t)} = 0.1213\frac{N_0}{r^3(t)} \tag{6.42}$$

$N_0 \simeq 1\times 10^8$ atoms start from the MOT in a ball of rms radius $r_{\text{MOT}} = 1.6$ mm (see Section 3.1.4). The rms velocity $v_{\text{MOT}} = 1.4$ cm/s causes the cloud radius to expand according to $r(t) = r_{\text{MOT}} + v_{\text{MOT}}t$. At the time of the late probe, the rms radius of the cold atom cloud is thus $r(0.52) = (0.16\,\text{cm}) + (1.4\,\text{cm/s})(0.52\,\text{s}) = 0.89$ cm,

[7] Actually, it is even smaller because as we will show later, at the time of the late probe the atom cloud is bigger than the probe beam, and thus the probe pulse does not detect all of the atoms in the cloud.

giving a density of $n_{\text{lateprobe}} \simeq 1.7 \times 10^7\,\text{cm}^{-3}$ using equation (6.42). This cold atom cloud density puts an upper limit on the room temperature cesium background of $n_{\text{hotbkgnd}} < (1.7 \times 10^7\,\text{cm}^{-3})/5.7 = 3.0 \times 10^6\,\text{cm}^{-3}$.

With this cesium density n_{hotbkgnd}, we can numerically evaluate the convolution function $F(\Delta)$ for each of the possible transitions. The presence of transitions to the D1 $F\!=\!3'$ excited state changes the laser wavenumber by a negligible fraction $\Delta k/k \simeq -1 \times 10^{-3}$ ppb/MHz, with no change when the lasers are perfectly on resonance. Even when the Raman lasers are detuned by up to ± 2 MHz to correct for the Doppler shift of the moving cold atom cloud, this contribution should still remain negligible. From the D1 $F\!=\!4'$ excited state the effect is roughly constant at $\Delta k/k \simeq +0.032$ ppb. Because the recoil measurement is proportional to k_{eff}^2, the potential measurement error is 0.064 ppb, still too small to be of concern. Since the D2 line is even farther detuned then the other D1 hyperfine excited state, the effect of the D2 transitions is also negligible.

6.5.2 Cold atom cloud

The cold atom cloud has a much smaller rms velocity. For $\sigma_v = 1.6$ cm/s (see Section 3.1.4) equation (6.40) is plotted in Figure 6.16 for the Raman lasers coupling to the D1 $F\!=\!3'$ excited state. Because of the coherence between the light fields and the atoms, in addition to the dependence on the single-photon detuning Δ, there will be finer dispersive features that depend on the two photon detuning δ. Instead of solving for the complete expression [56, 57], it is sufficient for our purposes here to simply estimate the slope of these features near resonance. The width of these two-photon features is determined by the effective Rabi frequency $\Omega_{\text{eff}} = \sqrt{\Omega_1^2 + \Omega_2^2}$ in equation (2.73) which should be compared with the width of the single-photon feature that scales roughly as the natural linewidth Γ. The slope of the two-photon dispersive features therefore should be approximately $-\Gamma/\Omega_{\text{eff}}$ times the ~ -83 ppb/MHz slope at resonance in Figure 6.16. With the effective Rabi frequency predicted in Appendix A.1.2 to be $\Omega_{\text{eff}} \simeq 2.5$ MHz, we estimate a slope of roughly$-(4.6\,\text{MHz})/(2.5\,\text{MHz})(-83\,\text{ppb/MHz}) = +153$ ppb/MHz for this higher

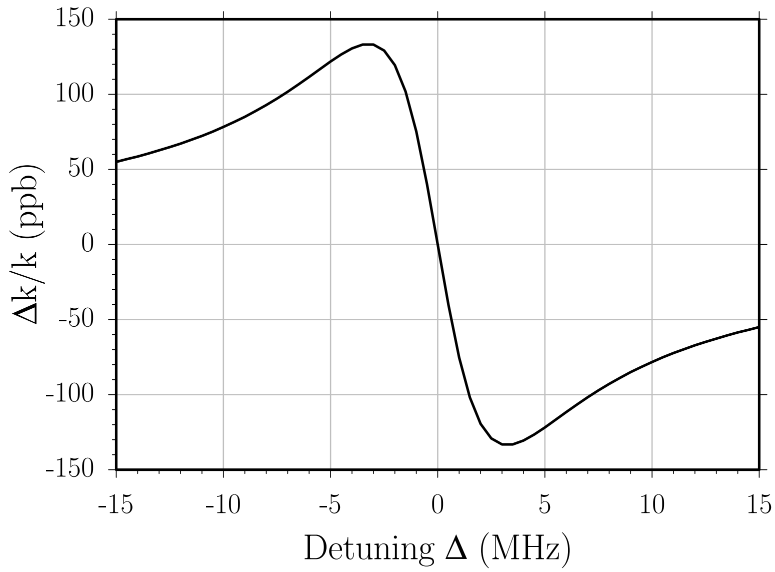


Figure 6.16: **Wavelength change** for a laser tuned to the $F=3'$ excited state interacting with a sample of cesium atoms whose rms velocity is 1.6 cm/s. We evaluate the convolution in equation (6.40) of the dispersive lineshape in equation (6.37) with the atomic velocity distribution in equation (6.39) for an atom density of $1 \times 10^8\,\mathrm{cm}^{-3}$.

order dispersion.

To determine how these dispersive features affect the final measurement, we now look more closely at the atoms in the cloud. Although the final signal comes from atoms in the $m_F=0$ states, there are atoms distributed throughout the other magnetic sublevels. Because of the non-zero magnetic bias field, these atoms are detuned from two-photon resonance by $\delta = (50\,\mathrm{kHz})m_F$, and thus they change the laser's wavefront spacing. However, because the initial distribution of atoms is symmetric about the $m_F=0$ state, the effect from the $m_F=+1$ atoms tends to cancel the wavelength shift from the $m_F=-1$ atoms, and so on. It is only because of the two-photon transfer pulses driven by $\hat{\sigma}_+$ polarized light that this symmetry is broken. For this reason, we neglect the initial distribution among magnetic sublevels and consider only the change in this distribution due to the adiabatic transfer pulses.

At the beginning of each adiabatic transfer pulse, all of the atoms in the cloud are projected into dark and bright states. The pulse transfers the subset of atoms $N_\mathrm{i}(\mathrm{DS0})$ in the initial $m_F=0$ dark state to the final state with efficiency ϵ_0, leaving

$N_\mathrm{f}(\mathrm{DS0}) = \epsilon_0 N_\mathrm{i}(\mathrm{DS0})$ atoms in the signal state. The number of atoms $N_\mathrm{i}(DS1)$ in the $m_F\!=\!1$ dark state are transferred with efficiency ϵ_1, and similarly for the $m_F\!=\!2$ dark state. We will neglect the fraction of atoms in the $m_F\!=\!+3$ because they are not coupled to the excited state via $\hat{\sigma}_+$-polarized light. The fraction of atoms in the $m_F\!<\!0$ states we also neglect because optical pumping with $\hat{\sigma}_+$-polarized light tends to move atoms out of these states. According to the result derived in equation (2.84), of the atoms in the $m_F\!=\!0$ bright state, a fraction $f_0 = 1/7$ fall back into the $m_F\!=\!0$ dark state. In the same way, one can derive the fractions $f_1 = 1/5$ and $f_2 = 5/19$ for the chance of the bright state atoms falling from the $F\!=\!3', m_F\!=\!+1'$ excited state back into the $m_F\!=\!1$ and $m_F\!=\!2$ dark states, respectively. Thus, after the adiabatic transfer pulse, the number of atoms in the $m_F\!=\!1$ and $m_F\!=\!2$ dark state is

$$\begin{aligned} N_\mathrm{f}(\mathrm{DS1}) &= \epsilon_1 N_\mathrm{i}(\mathrm{DS1}) + (1-\epsilon_0) f_1 N_\mathrm{i}(\mathrm{DS0}) \\ N_\mathrm{f}(\mathrm{DS2}) &= \epsilon_2 N_\mathrm{i}(\mathrm{DS2}) + (1-\epsilon_0) f_2 N_\mathrm{i}(\mathrm{DS0}) \end{aligned} \tag{6.43}$$

For these expressions, we have simplified the problem by 1) assuming that all of the atoms not transfered to the $m_F\!=\!0$ dark state spent some time in a bright state and 2) that the light fields define the same population ratios for the $m_F\!\neq\!0$ states as they do for the $m_F\!=\!0$ state[8]. The resulting atom number and densities are summarized in Table 6.1 for a representative interferometer example: $T = 120$ ms and $N = 30$.

Since the atoms in the $m_F\!=\!0$ state are on resonance, they do not contribute to the dispersion. Atoms that remain in the bright state are rapidly pumped to the $F\!=\!4, m_F\!=\!+4$ state which is not coupled to the excited state by the $\hat{\sigma}_+$ polarized Raman beams. The only atoms that might contribute to the dispersive effect are the atoms in the $m_F\!=\!+1$ and $m_F\!=\!+2$ states. From the 153 ppb/MHz slope the dispersive features, detunings of $\delta_1 = 50$ kHz and $\delta_2 = 100$ kHz, and the mean densities for a typical interferometer given in Table 6.1, we predict a change in the

[8] Because the ratio of the angular matrix elements (see Table A.2) coupling the $F\!=\!3, m_\mathsf{F}=0$ and $F\!=\!4, m_\mathsf{F}=0$ ground states to the $F\!=\!3', m_\mathsf{F}=+1'$ excited state are not the same as those for the $F\!=\!3, m_\mathsf{F}=+1$ and $F\!=\!4, m_\mathsf{F}=+1$ ground states, the $m_\mathsf{F}=1$ dark state will have different $F\!=\!3/F\!=\!4$ population ratios than will the $m_\mathsf{F}=0$ dark state, and similarly for the $m_\mathsf{F}=2$ dark state.

	Adiabatic Transfer Pulse							
	$[\pi]_{Sel}$	$[\pi/2]_1$	$[\pi/2]_2$	$[\pi]_1$	$[\pi]_{30}$	$[\pi/2]_3$	$[\pi/2]_4$	Mean
t	129.98	133.35	253.62	255.09	274.97	277.97	398.24	
V^{-1}	2.5129	2.4241	0.8500	0.8410	0.7309	0.7160	0.3475	0.8680
ϵ	0.5645	0.9380	0.4690	0.9380	0.9380	0.9380	0.4690	
N_0	0.5645	0.5295	0.2483	0.2329	0.0364	0.0341	0.0160	
n_0	1.4186	1.2836	0.2111	0.1959	0.0266	0.0244	0.0056	0.1575
N_0'	0.4355	0.0350	0.2812	0.0154	0.0024	0.0023	0.0181	0.0281
n_0'	0.1563	0.1588	0.1086	0.1037	0.0240	0.0227	0.0116	0.0608
n_1'	0.2189	0.1657	0.1041	0.0124	0.0004	0.0006	0.0015	0.0154
n_2'	0.2880	0.0967	0.0754	0.0034	0.0005	0.0005	0.0017	0.0145

Table 6.1: **Evolution of the atom density** for an interferometer with $T = 120$ ms and 30 π-pulses. The time in ms after the launch of the center of the velocity-selecting π-pulse, the $\pi/2$-pulses, and the first and last π-pulses are shown. Approximating the atom cloud's spatial distribution as a sphere, we use this time and the atoms' rms velocity to calculate the inverse of its volume V^{-1}. ϵ is the transfer efficiency of each pulse, including both the velocity selection factor for the velocity-selecting π-pulse and the factor of two projection loss from the second and fourth $\pi/2$-pulses. Accumulating this transfer efficiency, we calculate the fraction N_0 of the initial atoms in the $m_F = 0$ dark state. The density n_0 in cm^{-3} of this signal state is then N_0/V. We assume that $1 - \epsilon$ of the atoms entering every pulse are in the bright state with fraction N_0'. According to the different angular weighting factors, a fraction of these bright state atoms fall back into the $m_F = 0$, $m_F = +1$, and $m_F = +2$ dark states (see equation (2.84)). Once in a dark state, they remain there ready to be transfered by the next pulse. Because the $m_F \neq 0$ states are Zeeman shifted out of resonance, we modify the transfer efficiency for these states according to the two-photon resonance condition. Thus, at the end of each pulse, each of the dark state densities n_0', n_1', and n_2' has accumulated contributions from 1) atoms that began the pulse in the bright state but then fell into that dark state and 2) atoms that entered the pulse already in that dark state *and* were transfered by the pulse. Because the atoms in the signal state, including both the interfering (n_0) and phase randomized (n_0') densities, are resonant, they contribute negligibly to the total dispersion. It is the atom densities n_1' and n_2' that are detuned from resonance by 50 kHz and 100 kHz, respectively, that change the wavefront spacing of the Raman lasers.

lasers wavenumber of $\Delta k/k = 0.12$ ppb and 0.22 ppb, for the $m_F = +1$ and $m_F = +2$ states, respectively. Since the recoil frequency is proportional to k_{eff}^2, the total change $\Delta k/k = 0.34$ ppb must be doubled to arrive at predicted 0.68 ppb change in f_{rec} due to the dispersive effect of the cold atoms. Because of the complexity of predicting the exact error from this effect and because it is less than 1 ppb, we do not correct our final recoil but instead treat result as a rough estimate of the uncertainty due to dispersion.

The same process can be used to calculate the effect from transitions between the ground states and the D1 $F=4'$ excited state. Since both lasers are detuned by $\Delta \simeq -1.17$ GHz, there are no fine two-photon dispersive features. For a density of $1 \times 10^8\,\mathrm{cm}^{-3}$, we predict a $\Delta k/k = +1.0$ ppb relatively constant over the same range of detunings shown in Figure 6.16. Reducing this effect by the mean densities of the $m_F=+1$ and $m_F=+2$ states given in Table 6.1, we predict a change of 0.06 ppb in the value of f_{rec}, which is small enough to be neglected.

As an experimental verification that dispersion should not be a problem, we took recoil data while alternating on each launch between low and normal atom densities. By opening the mechanical trap shutter later than normal we could reduce the number of atoms launched by approximately a factor of 3.90 without significantly reducing the signal-to-noise ratio. The recoil value when using a reduced density cold atom cloud was shifted by $+7.3 \pm 10.5$ ppb, consistent with no effect. Assuming the dispersive effect is linearly proportional to the atomic density, this result gives a value of $-(7.3 \pm 10.5)/(1 - 1/3.90) = -9.8 \pm 14$ ppb for the change in the recoil value due to the dispersive effect from the cold atom densities we typically take data with.

6.6 Timing

This experiment is fundamentally a pulsed measurement. It samples points equally spaced in time defined by some oscillator. From these samples, we can calculate a value for the recoil shift. In the lab there are many sources of noise both random and periodic which may change the final measurement result. In general, random noise tends to make the result less precise but in the long term on average does not affect the final measurement value itself. Periodic noise, however, can systematically change the measurement results if each sample point occurs at exactly the same phase of this periodic disturbance. Besides reducing the magnitude of the noise, one can also minimize its effect by insuring that the sample points are not phase stable relative to any periodic disturbance. If the sample points occur irregularly along a sinusoidal signal, for instance, although the final signal may be noisier, on average the final measurement result will be unchanged. The two general types of noise we are concerned with here come from 60 Hz oscillations of the line voltage and any periodic fluctuations synchronized with the pulsed atomic fountain.

6.6.1 60 Hz line noise

The master trigger for the entire experiment comes from a *Stanford Research Systems* DG555 pulse generator. This device can be set to use its internal clock or the oscillations of the line voltage to restart itself. When the device is "line triggered", on every shot, the start trigger and consequently all of the different experimental stages occur at exactly the same phase relative to the 60 Hz line oscillations. If some 60 Hz noise from one of the power supplies were to perturb one of the phase sensitive devices such as the LORAN C timebase itself and the experiment were to sample this disturbance every time at the same point in its cycle, a systematic error could result.

To test for changes in the recoil measurement synchronized with the 60 Hz line signal, we temporarily installed another DG555 pulse generator set to trigger on the line voltage. This pulse generator produced a trigger signal which started the normal master trigger generator. By varying the time delay of this pre-trigger signal in time steps of $(1/60\,\mathrm{Hz})/6 = 2.78$ ms, we were able to change when the measurements occurred in 60° steps relative to the phase of the 60 Hz line signal. The results are shown in Figure 6.17. Both the peak-to-peak change and the sinusoidal fit amplitude are consistent with zero. The single-sine function may not model the actual disturbance very well, but line noise generally does not involve many higher harmonics. To be safe, we take data only when the master pulse generator is internally trigger by its internal oscillator, which is not phaselocked to the line signal. We also choose the repetition time of 0.908 s to be a non-integer multiple of 1/60 Hz: $(0.908\,\mathrm{s})(60\,\mathrm{Hz}) = 54\frac{12}{25}$ cycles.

6.6.2 Periodic fluctuations synchronized with launch

Any connections between the generation of the MOT and launch and the generation of the interferometers are indirect at best. The vibration isolation (VI) system, however, does serve as a mechanical connection to the phase of the Raman beams.

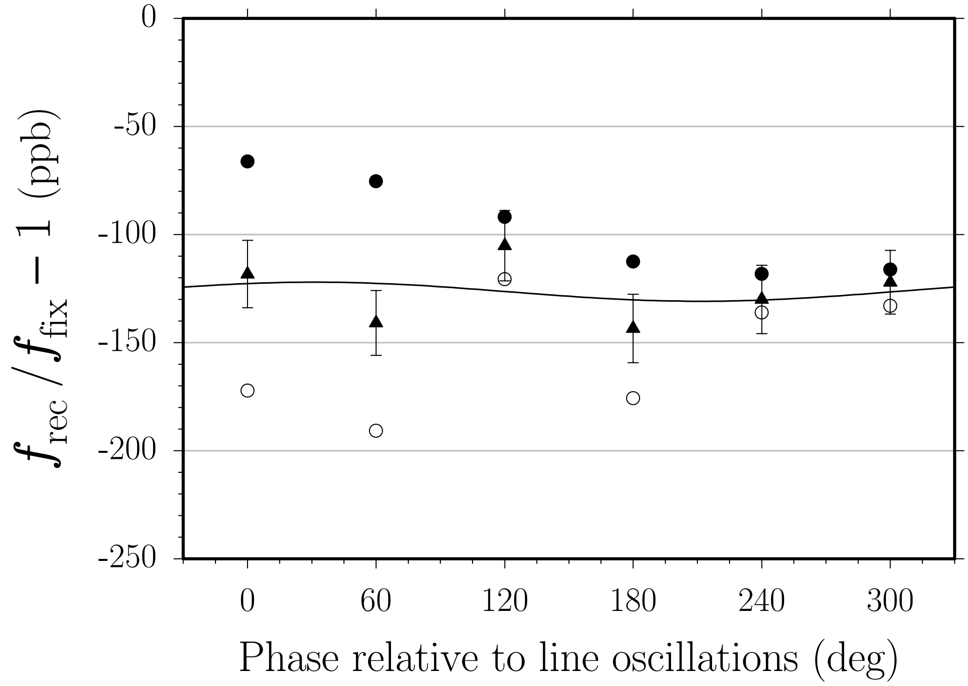


Figure 6.17: **Recoil frequency versus the phase of the 60 Hz line signal.** By triggering the master experiment trigger at fixed times relative to the 60 Hz line signal, we can verify that the final measurement value is negligibly affected by 60 fluctuations. The peak-to-peak variation of all of the points is 38 ± 23 ppb, consistent with no effect. Fitting these data, which consist of 126 repetitions of interferometers with $T = 120$ ms and 30 π-pulses, with a 60 Hz sinusoidal function gives an amplitude of 4.4 ± 10.6 ppb, a phase of $58 \pm 138°$, and an offset of -126.4 ± 7.5 ppb. Because the fit amplitude is consistent with zero and the fit phase is essentially undetermined, we conclude that the final measurement value has no sinusoidal dependence on 60 Hz line noise. For a detailed explanation of the graph symbols see [53].

The interferometer platform detects and corrects for any motion of the tracer beam wavefronts relative to the motion of the beamsplitter cube. In the limit that the beamsplitter cube accelerates uniformly with respect to the atoms, the atoms will observe perfectly steady wavefronts. If the cube moves, the tracer feedback loop will cause the Raman beam wavefronts to move in the same way. Random cube motion will manifest as increased measurement noise. If the cube moves in a systematic and repeatable way, however, the disturbance to the position of the laser wavefronts may not average out of the final interferometer phase. One possible way the experiment might disturb itself in this manner is via changing magnetic fields from either the MOT coils or the MOT trim coils. When the magnetic field generated by these coils changes, the magnetic force applied to any magnetic objects on the freely swinging VI column also changes. Such a mechanical kick applied via a magnetic force may not be completely removed by the vibration isolation feedback loop. Any remnant motion of the interferometer platform would be periodic and exactly synchronized with the measurement sampling. Thus, if there were a phase shift from this motion, it might not average to zero after many repetitions. Because internally the accelerometer of the VI system (see Section 3.4) uses magnetic force feedback to generate its output signal, it is also possible that the changing magnetic field might alter this output signal without even moving the VI column. If this happened, the sensor would falsely report some acceleration which the feedback loop would attempt to remove by driving the solenoid actuator and inappropriately moving the VI column. To minimize this particular effect, we installed a *Magnetic Shield Corp.* 35P70 magnetic shield around the accelerometer [46]. A final possible way through which the pulsing experiment might affect the motion of the VI column is via the mechanical shutters.

To test all of these possibilities, we observe the error signal from the closed loop VI system on an oscilloscope. With the experiment pulsing as normal and the oscilloscope triggered with the experiment, we can average the time trace and greatly reduce the amount of noise at frequencies not synchronized with the experiment pulsing at $1/(0.908\,\mathrm{s}) \simeq 1.10$ Hz. For a sinusoidal acceleration $a(t)$ given by

$$a(t) = a_0 \sin(\omega t) \tag{6.44}$$

the resulting velocity and position will be

$$v(t) = \frac{a_0}{\omega} \cos(\omega t) \tag{6.45}$$

$$z(t) = -\frac{a_0}{\omega^2} \sin(\omega t) \tag{6.46}$$

Since the interferometers have wavefront position sensitivity given by equation (2.11), the additional interferometer phase generated by this motion will be

$$\Delta\phi = k_{\mathrm{eff}}[z_1 - z_2 - z_3 + z_4]$$

$$= -k_{\text{eff}}\frac{a_0}{\omega^2}[\sin(\omega t_1) - \sin(\omega t_2) - \sin(\omega t_3) + \sin(\omega t_4)] \quad (6.47)$$

Since the time origin is arbitrary, $t_2 - t_1 = t_4 - t_3 = T$, and assuming the π-pulses take very little time $t_2 \simeq t_3$, equation (6.47) will have maximum absolute magnitude when the oscillation period $\frac{2\pi}{\omega} = 2T$. With appropriately chosen time origin and $\omega = \pi/T$, equation (6.47) will have magnitude

$$\begin{aligned}\Delta\phi &\simeq k_{\text{eff}}\frac{a_0}{(\pi/T)^2}[2+2] \\ &= \frac{4k_{\text{eff}}a_0T^2}{\pi^2}\end{aligned} \quad (6.48)$$

For $T = 120$ ms with 30 π-pulses a 0.7 mrad phase shift corresponds to a 1 ppb change in the recoil shift. Solving equation (6.48) for the acceleration amplitude a_0, we can determine an upper limit for how much the beamsplitter cube on the VI column would move in order to cause a 1 ppb shift in the final measurement

$$a_0 = \Delta\phi\frac{\pi^2}{4k_{\text{eff}}T^2} = (0.7\,\text{mrad})\frac{\pi^2}{4(2.24\times10^6\,1/\text{m})(0.12\,\text{s})} = 5.37\times10^{-8}\,\text{m/s}^2 \quad (6.49)$$

After averaging the VI closed loop error signal, the only response remaining comes from the vertically oriented MOT trim coils. The other MOT trim coils, the MOT coils themselves, and the mechanical shutters all produce no disturbance greater than $2\times10^{-9}\,\text{m/s}^2 = 0.04$ ppb. When the MOT trim coils switch, the vertical MOT trim coil produces a fast transient response of size $3\times10^{-8}\,\text{m/s}^2$ that has died out ($< 1\times10^{-8}\,\text{m/s}^2 \simeq 0.2$ ppb) by the time the interferometers occur.

As an addition verification that there is no systematic effect from some disturbance synchronized with the experiment's pulsing, we take data with and without the MOT trim coils pulsing and also with the same interferometer sequence starting at different times after the launch as shown in Figure 6.18. Both tests indicate no effect.

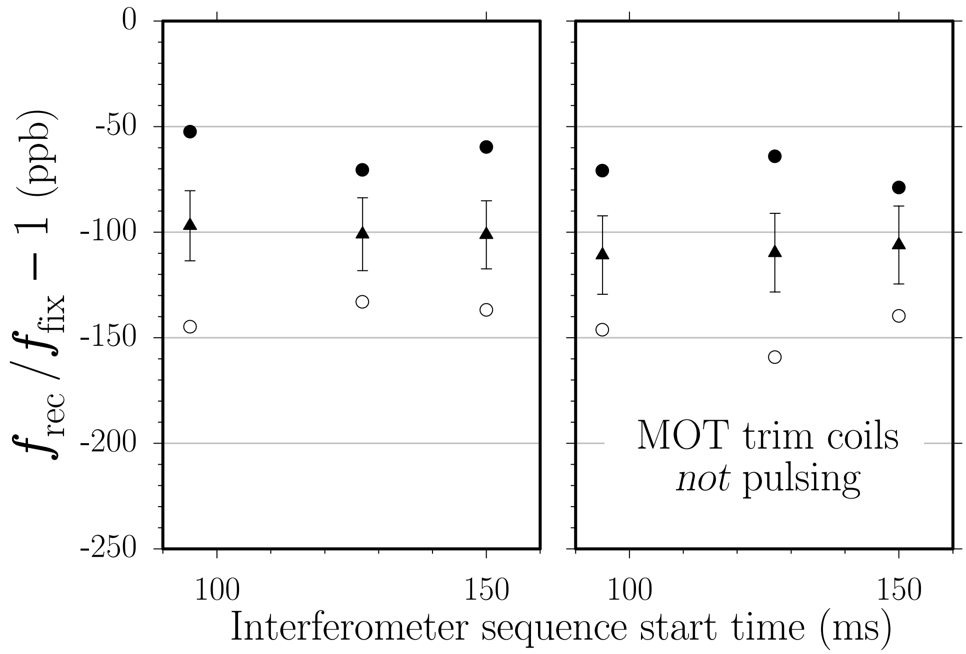


Figure 6.18: **Recoil frequency for the same interferometer sequence starting at different times in the fountain trajectory** comparing with (left) and without (right) the MOT trim coils pulsing. Both data sets were taken with $T = 120$ ms and 30 π-pulses. The final recoil data (solid triangles) on the left consisting of 48 repetitions have mean -108.8 ± 10.7 ppb and peak-to-peak variation of 4.3 ± 23.1. The data on the right (with the MOT trim coils *not* pulsing) consist of 49 repetitions and have mean -99.7 ± 9.6 ppb and peak-to-peak variation of 4.8 ± 26.2. The difference between the means of the two data sets is -9.0 ± 14.4 ppb, consistent with no difference. For a detailed explanation of the graph symbols see [53].

6.6.3 Time resolution

A further issue related to timing concerns the dynamic range required of the synthesizer which generates the pulse shapes. One of the most important systematic tests is to vary T (see Section 6.7), the time between the $\pi/2$-pulses. We vary T and thereby the total time for an interferometer $\sim 2T$ by almost two orders of magnitude. As mentioned in Section 3.6, because of finite memory size and processing speed, it is difficult to find a device that can synthesize patterns of these widely varying time lengths without changing the fundamental time resolution with which the pattern is synthesized. If the pulse shapes for interferometers with $T = 120$ ms and $T = 5$ ms were synthesized with different time resolutions, the pulse shapes might be different enough that we could not safely compare their results. With the gate synthesizer detailed in Section 3.6, which turns off the waveform synthesis clock when the light is off, the exact same pulse shapes can be distributed over much larger time intervals.

What happens when the time between the first two $\pi/2$-pulses $(t_2 - t_1)$ is not identical to the time between the final two $\pi/2$-pulses $(t_4 - t_3)$ is also an important yet somewhat elusive issue. If the recoil from the first three $\pi/2$-pulses is the same, then the two interferometer paths will not intersect at the fourth $\pi/2$-pulse. In order to interfere, the two interferometer paths must overlap in all aspects: momentum, space, internal state, *etc.*, so in principle, since the paths do not intersect in space, there will be no interference. However, because the individual atomic wavefunctions are spread in both position and momentum space and because the light pulses are not instantaneous in time, the areas of overlap are not sharp. From the temperature of the atoms launched from the MOT (see Section 3.1.4), according to the Heisenberg limit, the spatial spread of the individual atomic wavefunctions can be no narrower then roughly $\lambda/4$. At a relative speed of one two-photon recoil v_r, the timing discrepancy would have to be more than $\lambda/(4v_\mathrm{r}) = 32\,\mu$s for the interferometer paths to miss by more than the spatial spread of the individual atomic wavefunctions. This should not be a problem since the timing electronics driven mostly by 74LS- and 74F-series TTL logic sets up on the order of a few ns. However, even though the two paths interfere there can still be a phase error from some systematic timing discrepancy. In fact, the phase error should scale roughly as $k\Delta z = kv_\mathrm{r}\Delta t$, where Δt is the systematic timing error. Fortunately, because the light pulses for the up and down conjugate interferometers are driven in the same way by the same electronics, any timing error should be exactly the same for the up and down interferometers and thus cancel completely.

6.7 Adiabatic transfer

Because of its better π-pulse transfer efficiency, using adiabatic dark state transfer to impart recoils has made this experiment possible. Where adiabatic transfer makes things more difficult is with the $\pi/2$-pulses. The four $\pi/2$-pulse interferometer using adiabatic transfer depends on a high degree of symmetry between the $\pi/2$-pulses:

time reversal between the $\pi/2$-pulse pairs, intensity balance between the two light fields, simultaneity for the pulse edges, similarity between the pulse shapes of one interferometer compared to the pulse shapes of its conjugate, *etc.* Any net systematic phase shift from the $\pi/2$-pulses that does not cancel will change the measurement result. Many of these potential problems have been discussed in other sections, but because of the complexity of this issue, we have developed a technique for detecting and simultaneously removing all errors from the $\pi/2$-pulses. As the $\pi/2$-pulses are put closer together in time, the final interferometer phase depends less on the atom's spatial and temporal evolution than it does on some fixed phase error from the $\pi/2$-pulses. In fact, according to equation (2.34), in the limit as $T \to 0$, the final phase difference should vanish, because there is no time for the atomic superposition state to evolve in the dark at a rate different from that of the local oscillator. If the final phase difference is not zero, it must be from some phase error $\phi_{\rm err}$ from the $\pi/2$-pulses. For the real experiment, equation (2.52) must read

$$\Delta\Phi = \Phi_{\boxed{1}} - \Phi_{\boxed{2}} = \Phi_{\boxed{12}} = -\phi_{\rm err} - 4\pi(N+1)(f_{\rm rec} - f_{\rm fix})T \qquad (6.50)$$

In order to make a measurement of the recoil frequency independent of $\phi_{\rm err}$, one first measures the final interferometer phase difference with $T = 0$ and then with large T. The final phase difference for large T will include both the desired dependence on $f_{\rm rec} - f_{\rm fix}$ and on the phase error $\phi_{\rm err}$ from the $\pi/2$-pulses. By subtracting the result from the first measurement when T was zero, $\phi_{\rm err}$ can be removed. An even better method is to take data at several values of T. According to equation (6.50), the data plotted as phase versus T should lie on a line whose slope is the desired measurement $f_{\rm rec} - f_{\rm fix}$ and whose $T = 0$-axis intercept is the systematic error $\phi_{\rm err}$ from the $\pi/2$-pulses. The beauty of this measurement approach is that it removes *all* systematic phase errors due to the $\pi/2$-pulses, including problems from systematic non-adiabaticities, rf-amplitude dependent phase shifts, and intensity imbalances. The only errors it does not remove are phase errors that depend on T. However, because the experiment uses the exact same pulse shapes independent of T (see Section 3.6), it is difficult to imagine why errors from the $\pi/2$-pulses would vary with

T. Only problems that occur when the light is off should depend on T. Thus, in a single stroke, albeit with slightly more data than originally anticipated, we measure and remove all of the systematic measurement errors from the $\pi/2$-pulses.

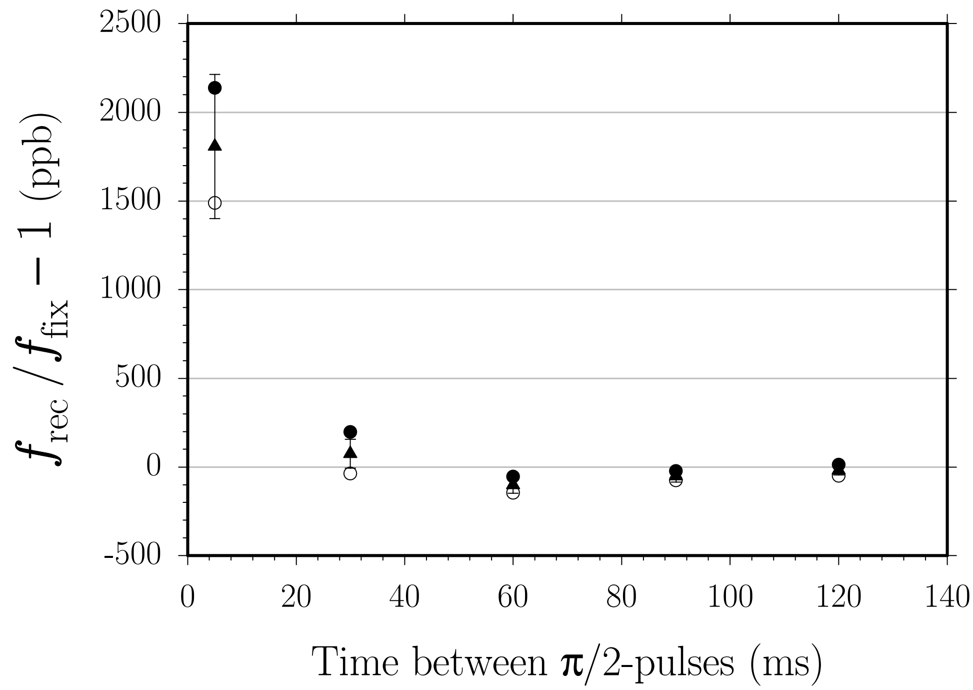


Figure 6.19: **Recoil frequency versus the time T between the $\pi/2$-pulses**, taken with 30 π-pulses. Because the interference fringe spacing is $1/T$, the final measurement resolution improves linearly with T. Thus, even though each recoil measurement (sold triangles) consists of roughly the same number of repetitions (129 in total), the uncertainty grows steadily larger as T shrinks. In fact, in terms of determining f_{rec} from this graph, only the last value with $T = 120$ ms contributes. For a detailed explanation of the graph symbols see [53].

To illustrate this process, consider the data shown in Figure 6.19 where we have varied the value of T. As T shrinks the measurement uncertainty in f_{rec} increases, because the measurement sensitivity increases linearly with T. Even with the large error bar, the value 1808 ± 406 ppb for $T = 5$ ms appears dramatically inconsistent with the other values. One might simply discard that value and use the others.

However, as shown in Figure 6.20a, when the same data are converted from frequency to phase by multiplying by $4\pi(N+1)T$, the systematic error is evident. The data all fall on a line, but the intercept which represents ϕ_{err} is not zero. The slope of this line divided by $4\pi(N+1)$ is the desired measurement Δf_{rec}, which is independent

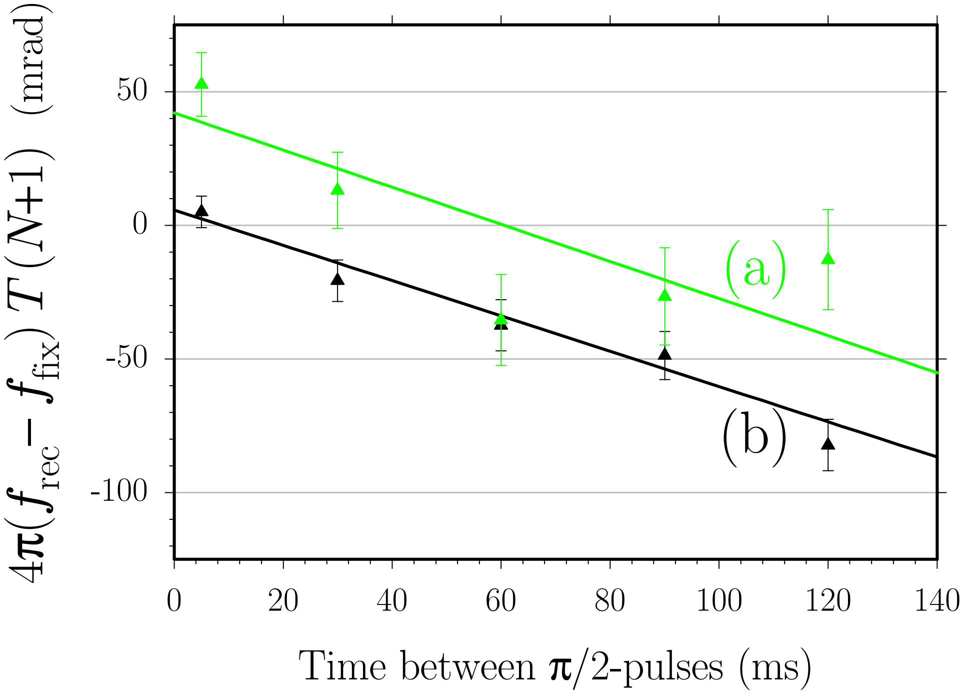


Figure 6.20: **Up/Down interferometer phase difference versus the time T between the $\pi/2$-pulses**, taken with 30 π-pulses. The points (a) shown in gray are the same data as shown in Figure 6.19 but with the vertical axis converted to phase by multiplying by $4\pi(N+1)T$, where T is the same horizontal axis. Note that unlike Figure 6.19, after converting to phase, all points have roughly the same uncertainty. By fitting each set of five points with a line, we can extract a value for $f_{\rm rec}$ and value for a systematic phase error $\phi_{\rm err}$ according to $\Delta\phi = \phi_{\rm err} + 4\pi(N+1)(f_{\rm rec} - f_{\rm fix})T$ (see equation (6.50)). For data set (a) we have $\phi_{\rm err} = 42.1 \pm 10.5$ mrad for the intercept and $f_{\rm rec} = -118.8 \pm 28.8$ ppb relative to $f_{\rm fix}$ from the slope. For data set (b) shown in black, which consists of 242 repetitions, we switch the $F=3$ and $F=4$ light fields on and off exactly together using the common switch AOM (see Section 3.2.4) instead of depending on the individual shaping AOMs to switch identically as in (a). The linear fit results for (b) are $\phi_{\rm err} = 4.9 \pm 5.1$ mrad for the intercept and $f_{\rm rec} = -111.8 \pm 12.4$ ppb relative to $f_{\rm fix}$ from the slope. Note that the systematic phase shift is no longer present when we use the common switch AOM. Also note that in spite of the significant systematic phase shift in data (a), within their uncertainties, we extract the same value for the recoil frequency as we do from (b). Thus, this method of converting to phase and fitting with a line removes the effect of the systematic phase error $\phi_{\rm err}$ from the data.

of the systematic phase error from the $\pi/2$-pulses.

When we discovered this systematic error and took more data (see Figure 7.1) to verify that it repeated, we spent a great deal of time and effort attempting to discover its source and remove it. Even though we were fairly confident we could measure and subtract it out using the "ϕ versus T" method, our concern was that because we did not understand the source of the systematic error, it might be causing other problems we were not aware of and were not properly controlling. We knew the problem had to come from the $\pi/2$-pulses, and we suspected that it came from the "phase-sensitive" edges of the pulses when both light fields turn on or off together to define the superposition dark state. We reduced the variation in phase from the variable rf attenuators used to shape the pulses (see Section 4.2), but the problem persisted. Finally, by introducing the common switch AOM, we were able to remove the problem and demonstrate that it did originate in the phase-sensitive edges. Figure 6.20a was taken with the phase-sensitive edges switched using the shaping AOMs instead the common switch AOM. Figure 6.20b is the exact same experimental condition except that the phase-sensitive edges of the $\pi/2$-pulses are switched on and off using the common switch AOM and not the shaping AOMs. Notice that the $T = 0$ intercept corresponding to ϕ_{err} is consistent with zero and disagrees with the intercept in Figure 6.20a by more than 3 standard deviations.

6.8 Interferometers

6.8.1 Sloping background

Whenever sinusoidal fringes are superimposed on a sloping background, the zero phase parameter from the fit may shift. If the background changes by a fraction s in units of Hz^{-1}, the phase zero point will shift by approximately $s/(2\pi TC)$, where C is the contrast of the fringes. In Section 6.2.2, we estimated that the maximum error we could have in the two-photon difference frequency is on the order of $\delta \sim 1$ Hz. The first derivative of the normalized lineshape function in equation (3.22) for an adiabatic transfer pulse is

$$\begin{aligned}\frac{\partial g}{\partial \delta} &= \frac{2\sqrt{\ln(2)}}{\delta_0\sqrt{\pi}}\left(-8\ln(2)\frac{\delta}{\delta_0^2}\right)\exp\left(-4\ln(2)\frac{\delta^2}{\delta_0^2}\right)\\ &= -\frac{16\ln(2)^{3/2}}{\sqrt{\pi}}\left(\frac{\delta}{\delta_0^3}\right)\exp\left(-4\ln(2)\frac{\delta^2}{\delta_0^2}\right)\end{aligned}$$

$$\simeq -\frac{16\ln(2)^{3/2}}{\sqrt{\pi}}\left(\frac{\delta}{\delta_0^3}\right) \tag{6.51}$$

when $\delta \ll \delta_0$. Our $\pi/2$-pulses have a FWHM linewidth of $\delta_0 = 95$ kHz, so the shift of the phase zero point will be

$$\begin{aligned} \Delta\phi &= -\frac{16\ln(2)^{3/2}}{\sqrt{\pi}}\left(\frac{\delta}{\delta_0^3}\right)\frac{1}{2\pi TC} \\ &= -15.2\frac{1\,\mathrm{Hz}}{95\,\mathrm{kHz}}\frac{1}{2\pi(120\,\mathrm{ms})0.15} = 79\,\mu\mathrm{rad} \end{aligned} \tag{6.52}$$

where we have assumed a conservative contrast of $C = 0.15$. If this slope were the same for the up and down interferometers, the phase shift cancels. Even if it does not cancel, however, it corresponds to a negligibly small error in the recoil frequency.

6.8.2 Fit routines and numerology

As discussed in Section 5.1, the fit routines use the Levenberg-Marquardt root finding method to minimize χ^2 the sum of the square differences between the fit function f_{fit} and the data $[\phi_i, a_i]$.

$$\chi^2 = \frac{1}{n-3}\sum_{i=1}^{n}\left[a_i - f_{\mathrm{fit}}(\phi_i)\right]^2 \tag{6.53}$$

where n is the number of data points. After determining the three fit parameters which minimize χ^2, the fit program checks each data point to see how it obeys the relation

$$\frac{\left|a_i - f_{\mathrm{fit}}(\phi_i)\right|}{\chi^2} \overset{?}{>} 3 \tag{6.54}$$

The program throws out all of the n_{bad} points that satisfy relation (6.54) and then refits the remaining $n - n_{\mathrm{bad}}$ data points. These "3σ-points" which occur roughly 2% of the time are usually due to some failure in at least one of the loading, launch, interferometer, or detection stages of the data taking process. Because this least-squares method places more importance on points lying many standard deviations away from the fit function, it is important that we separate out data points that do not obey normally distributed statistics. Per convention, we choose three standard deviations as the rejection threshold. A normal distribution expects just over two 3σ-points in a sample of 1000 points, our complete data set comprises over 10 000 sets of 51 point fringes, so we feel the threshold is justified.

As an irrefutable verification that the fit routines are doing what they are supposed to, we tested them on some fabricated data. Andreas Wicht generated simulated noiseless fringes with a shift that he chose but did not share with me. I ran these fake fringe data through the fit code as I normally would, and demonstrated that I could consistently reproduce his original value. We also tested the data analysis process on a set of simulated recoil data points with noise added to simulate the random measurement noise in the actual experiment. This noise was generated by repeated calls to a random number generator that outputs values normally distributed with a given standard deviation around a given mean. By choosing reasonable standard deviations for the parameters $A \simeq 0.0805 \pm 0.0026$, $C \simeq 0.1675 \pm 0.0137$, and $D \simeq 0 \pm 103$ mrad of equation (5.1), we generated over 3000 fake data files each containing four 50 point fringe scans, one for each interferometer. Using the fit code given in Appendix C.2, we fit these data exactly as we would for real data, and then compiled statistics. The resulting weighted mean differed from the original value used to generate the data by 0.60 ppb with a 1.06 ppb uncertainty. We therefore conclude that our data analysis process is trustworthy down to at least the one pbb level.

6.8.3 Missed recoils

A crucial ingredient to the precision of our measurement is the addition of N π-pulses in between the two sets of $\pi/2$-pulses. We demand that all of the atoms receive exactly N additional two-photon recoils. A phase error results if some of the atoms miss one or more momentum changing π-pulses. If an atom were to miss a π-pulse, it would be in the bright state at the beginning of the next pulse. This next π-pulse would then immediately drive single-photon transitions to the excited state from where the atom would spontaneously emit a photon and fall back to one of the ground states with randomized phase. On average these atoms will produce no net shift of the final interferometer phase. Further, because of the finite frequency width of a π-pulse, an atom missing several momentum transfers would be far enough off-resonance that it would no longer be addressed by succeeding π-pulses.

The most serious concern is that some of the atoms may experience a Doppler-free transition induced by two copropagating beams. To minimize the amount of copropagating light, we tilt all optics after and including the final polarizers away from normal incidence so that no back reflected light can illuminate the atoms. Besides the final polarizing beamsplitter cube, these optics include the $\lambda/4$-plates and the top and bottom windows of the vacuum chamber, which are mounted at a 5° angle. We also avoid applying π-pulses when the atoms are close to the top of their trajectory where the resonance conditions for Doppler-sensitive and Doppler-free transitions are degenerate. An atom that misses one momentum impulse will produce a final interferometer phase shifted by an amount $\Delta\phi = 2\pi f_{\mathrm{rec}} T$. We can slightly modify the time T so that $\Delta\phi$ is an integer multiple of 2π. With this choice of T, the fringe pattern for a single missed recoil will be the same as the fringe pattern for no missed recoils and will thus produce no net phase shift. If the fraction of atoms that miss a recoil is small, then the fraction of atoms that miss two recoils is even smaller, and so on. By canceling the effect of missing one recoil we remove the vast majority of any phase shift from missed recoils.

As a check on our sensitivity to missed recoils, we look for this systematic effect by adding 30 π-pulses at the top of the atomic trajectory so that the lasers are tuned near the Doppler-free resonance. We then scan the time T over a range where $\Delta\phi$ changes by 2π. The resulting fit to a single sinusoid has an amplitude that is consistent with zero. Normally, we set the time for the π-pulses so that they always occur before the top of the trajectory when their resonance frequency is almost 4 π-pulse line halfwidths from the Doppler-free resonance[9]. Thus, in addition to the

[9] This is only strictly true for the up interferometers that use the π-pulses to cancel the effect of gravity. For the down interferometers, the π-pulses push the atoms downward with an effective acceleration of $\sim 2g$. These atoms thus pass rapidly through the $v = 0$ point where Doppler-free resonances are not suppressed. However, because these atoms are accelerated so rapidly downward, no more than a few π-pulses are very close to the Doppler-free resonance.

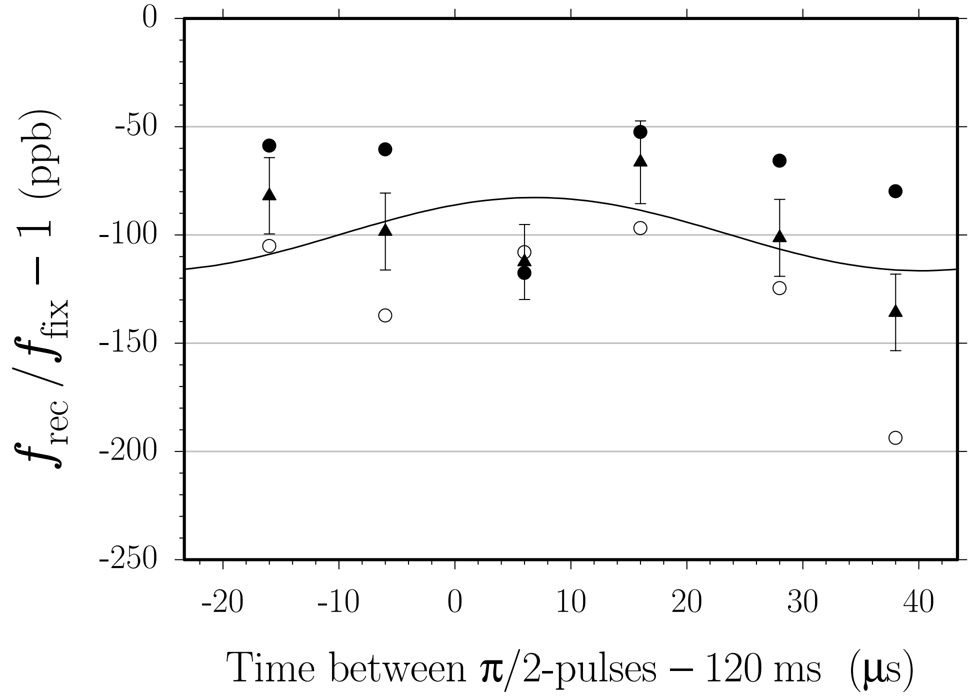


Figure 6.21: **Looking for missed recoils** with 30 π-pulses placed at the exact top of the trajectory when the resonance conditions for Doppler-free and Doppler-sensitive two-photon transitions are degenerate. Atoms that undergo a Doppler-free transition gain no momentum from that π-pulse. Consequently, the interferometer phase they produce is shifted by $\Delta\phi = 2\pi f_{\rm rec}T$. In this test consisting of 66 repetitions, we varied T around 120 ms in small 60° steps of $1/(6f_{\rm rec}) \simeq 11\,\mu$s to look for the presence of atoms missing one recoil, which would manifest itself as a sinusoidal fluctuation with a period of $1/f_{\rm rec}$. Fitting the data with a single sinusoid of this period gives an amplitude of 16.9 ± 16.5 ppb, a fit phase of $53 \pm 58°$, and an offset of -99.6 ± 12.1 ppb. Since the amplitude is consistent with zero and the phase is barely determined for an experimental condition that is much more sensitive than normal to Doppler-free transitions, we conclude that the final measurement value is not affected by missed recoils. For a detailed explanation of the graph symbols see [53].

suppression achieved from appropriately choosing a special value for T, we expect additional reduction of at least a factor of 100 because of the detuning from the Doppler-free resonance.

6.8.4 Sagnac effect

When an interferometer enclosing some non-zero spatial area A rotates with angular velocity $\mathbf{\Omega}$ in the plane of $\mathbf{A}$ relative to some absolute coordinate system, the final interferometer phase difference changes by

$$\Delta\phi = 2\frac{m}{\hbar}\mathbf{\Omega}\cdot\mathbf{A} \tag{6.55}$$

where m is the mass of the interfering particle [12]. Although some atom interferometer experiments are designed entirely to measure this very effect [15], for our experiment it is an undesired systematic error. The Earth rotates with angular velocity

$$|\mathbf{\Omega}_{\rm E}| = \Omega_{\rm E} \simeq \frac{2\pi}{(24\,{\rm h})(3600\,{\rm s/h})}\left(1-\frac{1}{365.24\,{\rm days/year}}\right) = 72.523\,\mu{\rm rad/s} \tag{6.56}$$

In our labs at Stanford located at $\theta = 37°25'44.4'' = 0.653\,2593$ rad latitude, the earth's angular rotation vector is

$$\begin{aligned}\mathbf{\Omega}_{\rm E} &= \Omega_{\rm E}(\cos\theta\,\hat{\mathbf{N}}S + \sin\theta\,\hat{\mathbf{U}}P) \\ &= (57.591\ \hat{\mathbf{N}}S + 44.078\ \hat{\mathbf{U}}P)\,\mu{\rm rad/s}\end{aligned} \tag{6.57}$$

Since without a great deal more federal funding, we cannot turn off this rotation, we must strive to minimize A and/or design the interferometers so that the effect cancels when the results from the four different interferometers are combined. Ideally, the atoms are launched vertically and the Raman beams are also aligned with $\mathbf{g}$. In this ideal case, the interferometers all take place on a vertical line, enclose no spatial area, and thus produce Sagnac phase.

Now consider the general case when the launch and the Raman beams are not vertical by angles $\theta_{\rm L}$ and $\theta_{\rm g}$, respectively. Assume they are both misaligned in the xz-plane. An example for the area opened with interferometer geometries $\boxed{1}$ and $\boxed{2}$ with no π-pulses when $\theta_{\rm L} > \theta_{\rm g} > 0$ is shown in Figure 6.22. To evaluate the area enclosed by each of these interferometers we write

$$\begin{aligned}A &= \Big|\,\frac{1}{2}(\mathbf{r}_{2\rm S}-\mathbf{r}_1)\times(\mathbf{r}_{2\rm U}-\mathbf{r}_1) \\ &\quad + (\mathbf{r}_{2\rm S}-\mathbf{r}_{2\rm U})\times(\mathbf{r}_{3\rm U}-\mathbf{r}_{2\rm U})\end{aligned} \tag{6.58}$$

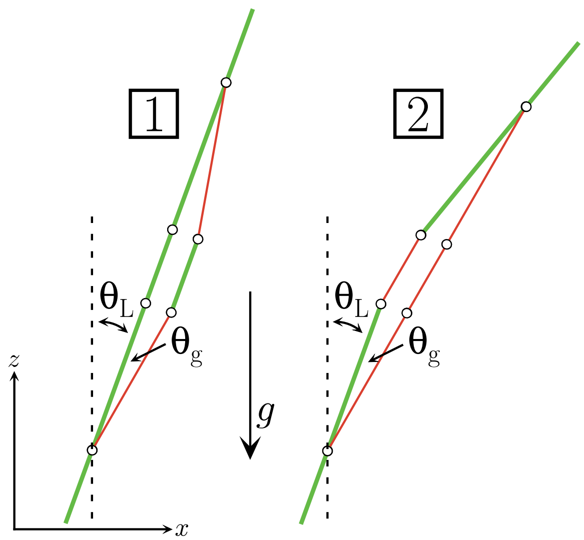


Figure 6.22: **Spatial area enclosed by the interferometers** due to misalignments in the initial launch direction and the verticality of the Raman beams. The initial launch direction and the Raman beams are misaligned from vertical by θ_L and θ_g, respectively. Here we have assumed that both misalignments are in the xz-plane and without loss of generality that $\theta_\mathrm{L} > \theta_\mathrm{g} > 0$. In the text we show that the areas enclosed by both interferometer geometries 1 and 2 are equal.

$$+ \frac{1}{2}(\mathbf{r}_4 - \mathbf{r}_{3\mathrm{U}}) \times (\mathbf{r}_4 - \mathbf{r}_{3\mathrm{S}}) \Big|$$

where $\mathbf{r}_{i\mathrm{S}} = x_{i\mathrm{S}}\,\hat{\mathbf{x}} + z_{i\mathrm{S}}\,\hat{\mathbf{z}} = (x_{i\mathrm{S}}, z_{i\mathrm{S}})$ is the two-dimensional position vector of the atoms at the ith $\pi/2$-pulse along the shifted path, and similarly for the unshifted path. Each line of equation (6.58) is the area from each one of the three intervals $(t_1 \to t_2, t_2 \to t_3, t_3 \to t_4)$ of the four $\pi/2$-pulse interferometers. Since $|\mathbf{r}_1 \times \mathbf{r}_2| = |x_1 z_2 - x_2 z_1|$, this equation becomes

$$\begin{aligned} A \;=\; \Big| \; & \frac{1}{2}\Big[(x_{2\mathrm{S}} - x_1)(z_{2\mathrm{U}} - z_1) - (z_{2\mathrm{S}} - z_1)(x_{2\mathrm{U}} - x_1)\Big] \\ & + \Big[(x_{2\mathrm{S}} - x_{2\mathrm{U}})(z_{3\mathrm{U}} - z_{2\mathrm{U}}) - (z_{2\mathrm{S}} - z_{2\mathrm{U}})(x_{3\mathrm{U}} - x_{2\mathrm{U}})\Big] \\ & + \frac{1}{2}\Big[(x_4 - x_{3\mathrm{U}})(z_4 - z_{3\mathrm{S}}) - (z_4 - z_{3\mathrm{U}})(x_4 - x_{3\mathrm{S}})\Big] \; \Big| \end{aligned} \tag{6.59}$$

We now evaluate all of the positions (x_i, z_i) throughout the interferometer, similar to equation (2.6) with the added complexity that the recoil direction and initial velocity are misaligned by angles $\theta_{\rm g}$ and $\theta_{\rm L}$. For the unshifted path,

$$z_{2\rm U} - z_1 = v_1 T\cos\theta_{\rm L} - \tfrac{1}{2}gT^2 \quad (6.60)$$
$$x_{2\rm U} - x_1 = v_{\rm L} T\sin\theta_{\rm L} \quad (6.61)$$
$$z_{3\rm U} - z_{2\rm U} = v_1 T'\cos\theta_{\rm L} - \tfrac{1}{2}g(2TT' + T'^2) \quad (6.62)$$
$$x_{3\rm U} - x_{2\rm U} = v_{\rm L} T'\sin\theta_{\rm L} \quad (6.63)$$
$$z_4 - z_{3\rm U} = v_1 T\cos\theta_{\rm L} - \tfrac{1}{2}g(3T^2 + 2TT') \quad (6.64)$$
$$x_4 - x_{3\rm U} = v_{\rm L} T\sin\theta_{\rm L} \quad (6.65)$$

and for the shifted path

$$z_{2\rm S} - z_1 = v_1 T\cos\theta_{\rm L} + v_{\rm r} T\cos\theta_{\rm g} - \tfrac{1}{2}gT^2 \quad (6.66)$$
$$x_{2\rm S} - x_1 = v_{\rm L} T\sin\theta_{\rm L} + v_{\rm r} T\sin\theta_{\rm g} \quad (6.67)$$
$$z_{3\rm S} - z_{2\rm S} = v_1 T'\cos\theta_{\rm L} - \tfrac{1}{2}g(2TT' + T'^2) \quad (6.68)$$
$$x_{3\rm S} - x_{2\rm S} = v_{\rm L} T'\sin\theta_{\rm L} \quad (6.69)$$
$$z_4 - z_{3\rm S} = v_1 T\cos\theta_{\rm L} - v_{\rm r} T\cos\theta_{\rm g} - \tfrac{1}{2}g(3T^2 + 2TT') \quad (6.70)$$
$$x_4 - x_{3\rm S} = v_{\rm L} T\sin\theta_{\rm L} - v_{\rm r} T\sin\theta_{\rm g} \quad (6.71)$$

Note that from equations (6.60), (6.61), (6.66), and (6.67)

$$z_{2\rm S} - z_{2\rm U} = v_{\rm r} T\cos\theta_{\rm g} \quad (6.72)$$
$$x_{2\rm S} - x_{2\rm U} = v_{\rm r} T\sin\theta_{\rm g} \quad (6.73)$$

In equations (6.60) through (6.71), we have left out the effect of the π-pulses, since the π-pulses shift both interferometer paths identically and thus they do not open up any area. Combining these equations into equation (6.59), the area A_{12} of the first time interval is

$$A_{12} = \Bigg| \frac{1}{2}\Big[(v_{\rm L} T\sin\theta_{\rm L} + v_{\rm r} T\sin\theta_{\rm g})\,(v_1 T\cos\theta_{\rm L} - \tfrac{1}{2}gT^2)$$

$$
\begin{aligned}
&\quad - \left(v_1 T\cos\theta_{\rm L} + v_{\rm r} T\cos\theta_{\rm g} - \tfrac{1}{2} g T^2\right)\left(v_{\rm L} T \sin\theta_{\rm L}\right)\Big]\Big| \\
&= \frac{1}{2}\left| v_{\rm r} T^2 (v_1 \sin\theta_{\rm g}\cos\theta_{\rm L} - v_{\rm L}\cos\theta_{\rm g}\sin\theta_{\rm L}) - \tfrac{1}{2} v_{\rm r} g T^3 \sin\theta_{\rm g}\right| \qquad (6.74)
\end{aligned}
$$

and similarly, the remaining regions have areas

$$
\begin{aligned}
A_{23} &= \Big| v_{\rm r} T T' (v_1 \sin\theta_{\rm g}\cos\theta_{\rm L} - v_{\rm L}\cos\theta_{\rm g}\sin\theta_{\rm L}) \\
&\qquad - \tfrac{1}{2} v_{\rm r} g T (2TT' + T'^2)\sin\theta_{\rm g}\Big| \qquad (6.75) \\
A_{34} &= \frac{1}{2}\Big| v_{\rm r} T^2 (v_1 \sin\theta_{\rm g}\cos\theta_{\rm L} - v_{\rm L}\cos\theta_{\rm g}\sin\theta_{\rm L}) \\
&\qquad - \tfrac{1}{2} v_{\rm r} g T (3T^2 + 2TT')\sin\theta_{\rm g}\Big| \qquad (6.76)
\end{aligned}
$$

Combining equations (6.74) through (6.76), the total area $A = A_{12} + A_{23} + A_{34}$ is thus

$$
\begin{aligned}
A\left(\boxed{1}\right) &= v_{\rm r} T \Big[(T + T')(v_1 \sin\theta_{\rm g}\cos\theta_{\rm L} - v_{\rm L}\cos\theta_{\rm g}\sin\theta_{\rm L}) \\
&\qquad - \tfrac{1}{2} g (2T^2 + TT' + T'^2)\sin\theta_{\rm g}\Big] \qquad (6.77)
\end{aligned}
$$

which does vanish if $\theta_{\rm L} = \theta_{\rm g} = 0$.

For the two paths of the conjugate interferometer $\boxed{2}$, we have

$$
\begin{aligned}
z_{3\rm U} - z_{2\rm U} &= v_1 T'\cos\theta_{\rm L} + v_{\rm r} T'\cos\theta_{\rm g} - \tfrac{1}{2} g (2TT' + T'^2) \qquad (6.78) \\
x_{3\rm U} - x_{2\rm U} &= v_{\rm L} T'\sin\theta_{\rm L} + v_{\rm r} T'\sin\theta_{\rm g} \qquad (6.79) \\
z_4 - z_{3\rm U} &= v_1 T\cos\theta_{\rm L} + 2 v_{\rm r} T\cos\theta_{\rm g} - \tfrac{1}{2} g (3T^2 + 2TT') \qquad (6.80) \\
x_4 - x_{3\rm U} &= v_{\rm L} T\sin\theta_{\rm L} + 2 v_{\rm r} T\sin\theta_{\rm g} \qquad (6.81)
\end{aligned}
$$

and for the shifted path

$$
\begin{aligned}
z_{3\rm S} - z_{2\rm S} &= v_1 T'\cos\theta_{\rm L} + v_{\rm r} T'\cos\theta_{\rm g} - \tfrac{1}{2} g (2TT' + T'^2) \qquad (6.82) \\
x_{3\rm S} - x_{2\rm S} &= v_{\rm L} T'\sin\theta_{\rm L} + v_{\rm r} T'\sin\theta_{\rm g} \qquad (6.83)
\end{aligned}
$$

$$z_4 - z_{3S} = v_1 T\cos\theta_L + v_r T\cos\theta_g - \tfrac{1}{2}g(3T^2 + 2TT') \tag{6.84}$$

$$x_4 - x_{3S} = v_L T\sin\theta_L + v_r T\sin\theta_g \tag{6.85}$$

The expressions for the first time interval between the first two $\pi/2$-pulses have been omitted because they are identical to equations (6.60), (6.61), (6.66), and (6.67) for interferometer $\boxed{1}$. As a result, the area A_{12} given in equation (6.74) is also the same. Because equations (6.72) and (6.73) are also the same for interferometer $\boxed{2}$ the areas A_{23} and A_{34} for the remaining regions are

$$A_{23} = \Big| v_r TT'(v_1\sin\theta_g\cos\theta_L - v_L\cos\theta_g\sin\theta_L) - \tfrac{1}{2}v_r gT(2TT' + T'^2)\sin\theta_g\Big| \tag{6.86}$$

$$A_{34} = \frac{1}{2}\Big| v_r T^2(v_1\sin\theta_g\cos\theta_L - v_L\cos\theta_g\sin\theta_L) - \tfrac{1}{2}v_r gT(3T^2 + 2TT')\sin\theta_g\Big| \tag{6.87}$$

Combining equations (6.74), (6.86), and (6.87), the total area $A = A_{12} + A_{23} + A_{34}$ for this interferometer is

$$A\left(\boxed{2}\right) = v_r T\Big[(T + T')(v_1\sin\theta_g\cos\theta_L - v_L\cos\theta_g\sin\theta_L) - \tfrac{1}{2}g(2T^2 + TT' + T'^2)\sin\theta_g\Big] \tag{6.88}$$

identical to expression (6.77) for the total area enclosed by interferometer geometry $\boxed{1}$. For $\theta_L \ll 1$ and $\theta_g \ll 1$, this area becomes

$$A\left(\boxed{1}\right) = A\left(\boxed{2}\right) \simeq v_r T\Big[(T + T')(v_1\theta_g - v_L\theta_L) - \tfrac{1}{2}g(2T^2 + TT' + T'^2)\theta_g\Big] \tag{6.89}$$

independent of the number of π-pulses. Assuming $T = 135$ ms and $N = 30$ (implying $T' = 24$ ms), equation (6.89) becomes

$$A\left(\boxed{1}\right) = A\left(\boxed{2}\right)$$
$$\simeq (6.71\,\mathrm{mm/s})(135\,\mathrm{ms})\Big\{(135 + 24\,\mathrm{ms})\Big[(1.76\,\mathrm{m/s})\theta_g - (3.00\,\mathrm{m/s})\theta_L\Big]$$

$$
\begin{aligned}
& - \tfrac{1}{2}(9.80\,\mathrm{m/s^2})\left[2(135\,\mathrm{ms})^2 + (135\,\mathrm{ms})(24\,\mathrm{ms}) + (24\,\mathrm{ms})^2\right]\theta_\mathrm{g}\Big\} \\
= \; & (0.906\,\mathrm{mm})\left[(0.279\,\frac{\mathrm{mm}^2}{\mathrm{mrad}})\theta_\mathrm{g} - (0.477\,\mathrm{mm}^2/\mathrm{mrad})\theta_\mathrm{L} - (0.197\,\mathrm{mm}^2/\mathrm{mrad})\theta_\mathrm{g}\right] \\
= \; & (0.074\,\mathrm{mm}^2/\mathrm{mrad})\theta_\mathrm{g} - (0.432\,\mathrm{mm}^2/\mathrm{mrad})\theta_\mathrm{L} \qquad (6.90)
\end{aligned}
$$

We now insert upper limits for $|\theta_\mathrm{L}| \leq 0.3$ mrad and $|\theta_\mathrm{g}| \leq 0.5$ mrad from Section 6.1.1. As a worst case estimate, we further assume that the angles θ_L and θ_g are opposite in sign and open in the plane in which the Earth's rotation rate is non-zero. A conservative value for the exposed spatial area of the interferometers is then

$$
A\left(\boxed{1}\right) = A\left(\boxed{2}\right) = A = 0.179\,\mathrm{mm}^2 \qquad (6.91)
$$

which produces a Sagnac phase of

$$
\begin{aligned}
\Delta\phi &= 2\frac{m}{\hbar}\mathbf{\Omega}_\mathrm{E}\cdot\mathbf{A} = \frac{2}{v_\mathrm{r}\lambda_\mathrm{eff}}\Omega_\mathrm{E}(\mathrm{NS})A \\
&= \frac{2(57.591\,\mu\mathrm{rad/s})(0.179\,\mathrm{mm}^2)}{3.00\times10^{-9}\,\mathrm{m}^2/\mathrm{s}} = 6.87\,\mathrm{mrad} \qquad (6.92)
\end{aligned}
$$

To the extent that all of the variables in equations (6.77) and (6.88) are the same for both the up and down interferometers, because of the up/down subtraction, there will be no net contribution from the Sagnac effect. With many π-pulses, however, there is a mechanism through which the effective transverse launch velocity $v_\mathrm{L}\sin\theta_\mathrm{L} \simeq v_\mathrm{L}\theta_\mathrm{L}$ varies between the up and down interferometers. Because the π-pulses of $\boxed{2}$ push the atoms up, it takes longer for the atomic cloud longer to reach the detection region than it would with no π-pulses. For 30 π-pulses, the probe time must be delayed by $\sim$25 ms, or roughly $+25/520 = +5.2\%$. Analogously, the π-pulses of interferometer $\boxed{1}$ push the atoms down thereby advancing the probe time by the same amount. If the atoms are launched with some non-zero transverse velocity $v_\mathrm{L}\theta_\mathrm{L}$, they will travel farther off-axis for the up interferometers than they will for the down interferometers. Or conversely, since the detection region is fixed, the detection pulse after an up interferometer will favor a different initial transverse velocity or launch angle θ_L than

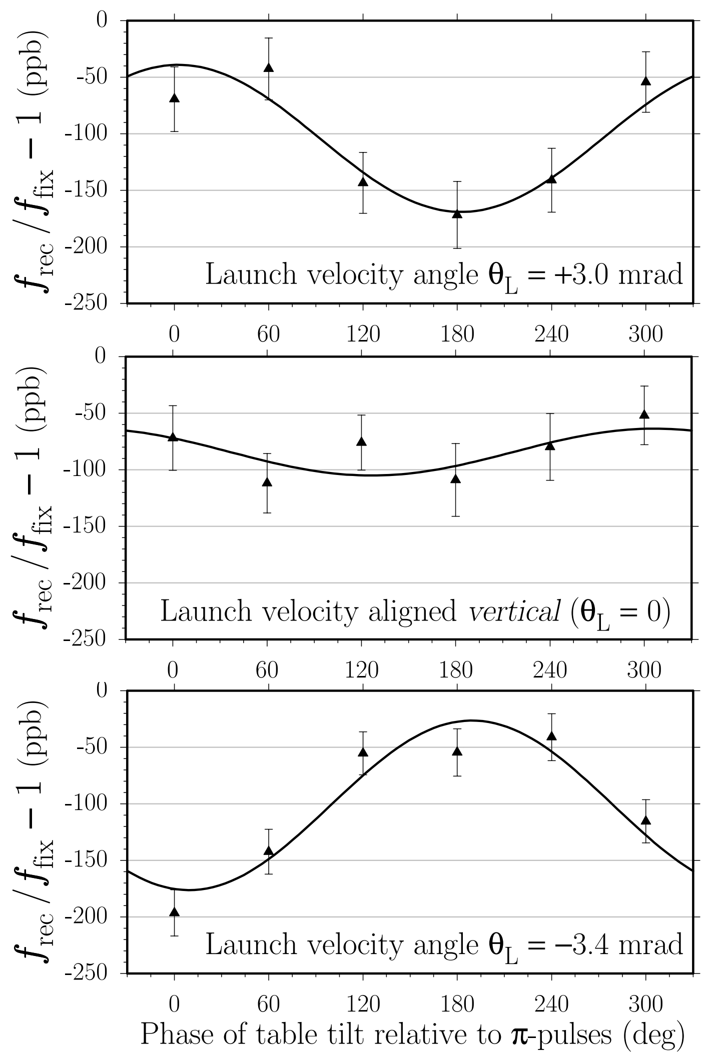


Figure 6.23: **Sagnac effect due to a misalignment of the launch velocity.** Recoil data are taken while the optical table tilts sinusoidally with amplitude 312 μrad and frequency 0.546 Hz about an East-West axis, equivalent to an angular velocity $\sim$18.6 times that of the Earth. We offset the time when the interferometers occur to six equally spaced points relative to a single period of this sinusoidal tilting. From sinusoidal fit amplitudes shown in following table, we see a clear effect when the launch velocity is misaligned by $\theta_L = +3.0$ mrad (top) and $\theta_L = -3.4$ mrad (bottom) and a result consistent with no effect when the launch is aligned vertical as normal (middle).

θ_L (mrad)	**Repetitions**	**Amplitude** (ppb)	**Phase** (deg)	**Offset** (ppb)
+3.0	84	65.0 ± 15.1	87.5 ± 13.6	-104.0 ± 10.8
0	96	20.7 ± 12.8	143.5 ± 34.0	-84.2 ± 8.8
−3.4	96	75.0 ± 14.6	-99.4 ± 11.4	-101.2 ± 10.4

the same pulse occuring 50 ms earlier for a down interferometer. For 30 π-pulses, the Sagnac effect due to a misaligned launch direction should cancel to only $|\{\theta_{\rm L}(\text{down}) - \theta_{\rm L}(\text{up})\}|/\theta_{\rm L} \simeq 10.4\%$. For this example, the up/down cancelation reduces the worst case estimation in equation (6.92), to 0.701 mrad equivalent to 0.89 ppb for $T =$ 135 ms. Because the inverted interferometers $\boxed{3}$ and $\boxed{4}$ exchange the roles of the up and down interferometers, for the exact same launch direction misalignment $\theta_{\rm L} \neq 0$, these interferometers will produce the same Sagnac phase but opposite in sign from the normal interferometers. Thus, this Sagnac effect will be further reduced down to the difference in probe times between the normal and inverted interferometers, which is $\sim$0.3% for interferometers with $T = 135$ ms and $N = 30$. Again, using the result from equation (6.92), we have an error of 0.021 mrad or 0.026 ppb.

To verify this insensitivity to rotations, we took recoil data after intentionally misaligning the launch and Raman beam directions. To additionally amplify the effect we installed "pusher legs"[10] on the optical table and rocked the table back and forth with a sinusoidal signal. By varying the amplitude and sign of the sinusoidal signal driving each of the pusher legs, we were able to produce rotational motion about arbitrary axes. To avoid convolving in the Earth's rotation, we programmed the legs to rotate about an East-West axis, orthogonal to the local component of the Earth's rotation. We further constrained the three pushing strengths so that the rotation axis was near the vibration isolation (VI) system, thus minimizing the amount of translational motion the VI system would have to detect and remove. With the table tilting sinusoidally with amplitude $\Theta_0 = 312\,\mu$rad and frequency $\omega/(2\pi) = 0.546$ Hz, we triggered the experiment at different points relative to this motion. At extreme

[10]The pusher legs were designed for use in the atom interferometry measurement of g (See [54, 26]). They are constructed from 10 inch outer diameter hollow aluminum cylinders. The cylinders are welded to hexagonal aluminum 1 inch thick base plates and filled with sand. The tops are made from 0.75 inch thick anodized aluminum optical breadboard with a clearance hole drilled in the center. The clearance hole just passes a 1.5 inch diameter stainless steel post that is held to the top plate by a *Thor Labs* post clamp attached to the underside of the top plate. On top of the 1.5 inch post is the 2.5 inch diameter solenoid part of an audio speaker. The permanent magnet part of the audio speaker is attached rigidly to the underside of the optical table. With the 1.5 inch post fully lowered, the pusher leg can be slid under the optical table so that it is centered under the permanent magnet. The 1.5 inch post is then raised until the coil completely vanishes in the permanent magnet. By measuring the response for different post heights, the position of the coil relative to the permanent magnetic can be optimized for maximum force.

tilt points, the rotational velocity is minimal but the table tilt is maximal. One quarter of a cycle later, the reverse is true and the angular velocity maximizes to $\omega\Theta_0 = (2\pi)(0.546\,\mathrm{Hz})(312\,\mu\mathrm{rad}) = 1070\,\mathrm{rad/s} \simeq 18.6\Omega_{\mathrm{E}}(\mathrm{NS})$. Recoil data taken with table tilting are shown in Figures 6.23 and 6.24. Six different values for the experiment trigger delay were chosen so that the center of interferometer π-pulses occurred at six evenly spaced points relative to the tilt oscillation. In Figure 6.23 the probe beam was displaced by -4.8, 0, and $+4.2$ mm in the North-South direction. With the probe displaced by Δx from center, the detection favors an atomic distribution whose initial transverse velocity is $\Delta x/t_{\mathrm{probe}} = (+4.0\,\mathrm{mm})/(0.52\,\mathrm{s}) = 8.1$ mm/s. Relative to an initial vertical launch velocity of 2.7 m/s, these probe positions correspond to initial launch direction angles of $\theta_{\mathrm{L}} = (8.1\,\mathrm{mm/s})/(2.7\,\mathrm{m/s}) = +3.0$ mrad (top), 0 mrad (middle), and -3.4 mrad (bottom). In Figure 6.24, the probe beam is set back to its center position, but the Raman beam direction tilted in the North-South direction by $\theta_{\mathrm{g}} = +2.8$ mrad (top), 0 mrad (middle), and -2.8 mrad (bottom). For each graph we fit the six points from each condition with the function

$$(\mathrm{Amplitude})\cos[\phi + (\mathrm{Phase})] + (\mathrm{Offset}) \tag{6.93}$$

Notice that when θ_{L} and θ_{g} change sign, the fit phase changes by $\sim 180\,\mathrm{deg}$, a clear indication that we are observing a real effect. The amplitude of the sine-wave fit when θ_{L} and θ_{g} are non-zero allows us to estimate the size of the effect when everything is aligned and the only tilt comes from the Earth's rotation. Because our laboratory angular velocity is ~ 18.6 times the size of the Earth's, we can reduce the estimate for the systematic by this factor. Thus, if we can make the launch vertical to within 0.3 mrad (see Section 6.1.1), this alignment uncertainty coupled with the Earth's rotation will causes a systematic error of 0.0 ± 0.3 ppb. If we can make the Raman beams vertical to within 0.5 mrad (see Section 6.1.1), the Sagnac shift this creates changes the final recoil measurement by at most 0.0 ± 1.0 ppb.

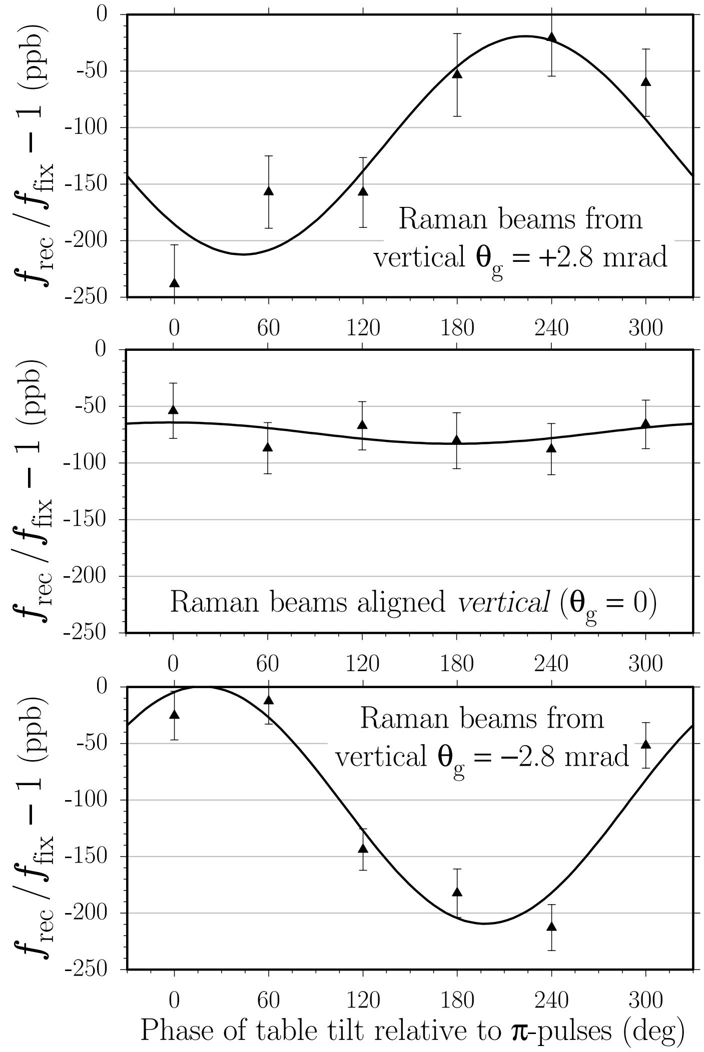


Figure 6.24: **Sagnac effect due to a non-verticality of the Raman beams.** Recoil data are taken while the optical table tilts sinusoidally with amplitude 312 μrad and frequency 0.546 Hz about an East-West axis, equivalent to an angular velocity ~18.6 times that of the Earth. We offset the time when the interferometers occur to six equally spaced points relative to a single period of this sinusoidal tilting. From sinusoidal fit amplitudes shown in following table, we see a clear effect when the Raman beams are misaligned by $\theta_g = +2.8$ mrad (top) and $\theta_g = -2.8$ mrad (bottom) and a result consistent with no effect when the beams is aligned vertical as normal (middle).

θ_g (mrad)	**Repetitions**	**Amplitude** (ppb)	**Phase** (deg)	**Offset** (ppb)
+2.8	114	96.6 ± 27.2	-133.6 ± 16.6	-115.6 ± 19.5
0	97	9.4 ± 8.4	91.7 ± 52.9	-73.5 ± 6.0
−2.8	116	105.0 ± 18.9	72.3 ± 10.6	-104.4 ± 13.6

6.9 Fundamental

6.9.1 Collisional shifts

In a dense gas of atoms, the mean field atom-atom interaction energy causes a shift Δf_{34} of the ground hyperfine splitting. This splitting causes a measurement error of $\Delta f_{34}/[(N+1)f_{\text{rec}}]$. This shift has been measured for the $F=3, m_F=0$ and $F=4, m_F=0$ states of cesium to be $\Delta f_{34} = -16$ mHz at an atomic density of $n = 1 \times 10^9\,\text{cm}^{-3}$ [58, 59]. Since all of the atoms in the cloud contribute to the collisional shift, we use just the expansion factor row (V^{-1}) in Table 6.1 to estimate a mean density of $\sim 1 \times 10^8\,\text{cm}^{-3}$. At this density, the collisional shift will be $\Delta f_{34} = -1.6$ mHz, leading to a measurement error of 3.4 ppb for interferometers with 30 $\pi/2$-pulses. This effect is proportional to the atomic density. The light pulses occur at the same time in the atom's trajectory for the up and down interferometers. Any variation in the atomic density throughout the interferometer should therefore be the same for each interferometer and its conjugate. Thus, this collisional shift which only changes the frequency between the hyperfine ground states should subtract out with the difference between the up and the down interferometers.

6.9.2 Relativity

All relativistic effects such as the second-order Doppler shift and time dilation effects will be proportional to $(v/c)^2$ where v is the atomic velocity and c is the speed of light. The largest atomic velocities during the interferometers are ~ 2 m/s, so these effects enter at the 10^{-7} ppb level and can therefore be neglected. Refer to [60, 61, 62] for relativistic calculations.

6.9.3 Gravitational red shift

As a photon moves within a gravitational potential its energy changes according to

$$U = G\frac{M_{\mathrm{E}} m_\gamma}{r} \tag{6.94}$$

where G is Newton's gravitational constant, M_{E} is the mass of the Earth, r is the distance between the photon and the Earth's center of mass, and $m_\gamma = \hbar\omega_0/c^2$ is the effective mass of the photon. To lowest order the fractional change in the photon's frequency $\Delta\omega/\omega$ is then

$$\frac{\Delta\omega}{\omega} = \frac{1}{\hbar\omega_0}\frac{\partial U}{\partial r}\bigg|_{R_{\mathrm{E}}}\Delta r = \frac{1}{\hbar\omega_0}\left(-\frac{GM_{\mathrm{E}}}{R_{\mathrm{E}}^2}\right)\frac{\hbar\omega_0}{c^2}\Delta r = \frac{g}{c^2}\Delta r \tag{6.95}$$

Even over the entire $\Delta r \simeq 30$ cm length of the magnetic shielding, this effect $\Delta\omega/\omega = (9.8\,\mathrm{m/s^2})/(3.0\times10^8\,\mathrm{m/s})^2(30\,\mathrm{cm}) = 3.2\times10^{-17}$ is too small to produce a significant measurement error.

Table 6.2: **Systematic Error Budget.** Each systematic effect is discussed in the section indicated in the first column. A limit in ppb from experimental tests, theoretical calculations, or both is given, followed by the correction applied to final value for f_{rec}. The length of the bar at the end of each row is proportional to the square of the uncertainty from each effect. These uncertainties are summed in quadrature to produce the total final systematic uncertainty.

	Systematic Effect	**Experiment Limit** (ppb)	**Theory Limit** (ppb)	**Correction to f_{rec}** (ppb)
Beams				
6.1	Guoy phase shift		-0.89 ± 0.04	$+0.89$
6.1.1	Wavefront curvature	0.05 ± 0.16	< 0.04	
6.1.2	Clipping	$+0.035 \pm 0.039$		
6.1.3	Speckle	< 0.1		
6.1.4	Relative angle	-1.0 ± 0.4		$+1.0$
6.1.5	Polarization	$\pm(1.5 \pm 2.0)$		
Frequencies				
6.2.1	Lock to cesium		< 0.6	
6.2.2	Difference frequency	< 0.002		
6.2.3	Difference frequency switching	< 0.4		
6.2.4	Gravity chirp		< 0.002	
6.2.5	Gravity gradient		$< 2 \times 10^{-7}$	
6.2.6	Bad Frequencies		0	
6.2.7	Computer arithmetic		0	
Electric Fields				
6.3.1	dc-Stark effect		$< 2 \times 10^{-4}$	
6.3.2	ac-Stark from tracer laser	< 0.004	< 0.008	
6.3.2	ac-Stark from Raman lasers	0.016 ± 0.10		

Table 6.3: **Systematic Error Budget continued.** From both tables, we find a total systematic correction of +2.74 ppb of f_{fix}, or +82.23 μHz. Summing the systematic uncertainties in quadrature gives a total systematic uncertainty of 3.23 ppb.

Systematic Effect		Experiment Limit (ppb)	Theory Limit (ppb)	Correction to f_{rec} (ppb)
Magnetic Fields				
6.4	Linear term	-1.0 ± 2.0		+1.0
6.4	Quadratic term	$+0.150 \pm 0.098$		−0.15
Dispersion				
6.5.2	Cold signal atoms	-9.8 ± 14	0	
6.5.2	Cold background atoms		< 0.7	
6.5.1	Hot background gas		< 0.06	
Timing				
6.6.1	Line noise		0	
6.6.2	Synchronized fluctuations		< 0.2	
6.6.3	Time resolution		0	
Adiabatic transfer				
6.7	$\pi/2$-pulses	See Figure 7.1		
Interferometers				
6.8.1	Sloping background		0	
6.8.2	Fit routines and numerology		0	
6.8.3	Missed recoils	< 0.02		
6.8.4	Sagnac effect from launch direction	< 0.3		
6.8.4	Sagnac effect from beam direction	< 1.0		
Fundamental				
6.9.1	Collisional shifts		< 0.3	
6.9.2	Relativity		$< 10^{-7}$	
6.9.3	Gravitational red shift		< 0.001	

Chapter 7

Determination of α

7.1 A final value for f_{rec}

Over the several years during which we ran this experiment, testing it for systematic problems, we accumulated a set $\{f_i, \sigma(f_i)\}$ of over 10 000 different measurements of f_{rec}. Each of these values for f_{rec} represents four measurements, one for each of the four different interferometer geometries discuss in Section 2.1. Of these data, we immediately reject the subset when we were amplifying some systematic effect beyond its normal operating level. For example, we reject the data we took when we changed the magnetic bias field or rocked the optical table to look for the Sagnac phase. From the remaining $n = 4000$ points, we can compute the weight mean

$$\langle f_{\text{rec}} \rangle = \frac{\displaystyle\sum_{i=1}^{n} \frac{f_i}{\sigma(f_i)^2}}{\displaystyle\sum_{i=1}^{n} \frac{1}{\sigma(f_i)^2}} \tag{7.1}$$

and the statistical uncertainty of the mean

$$\sigma(\langle f_{\text{rec}} \rangle) = \left(\sum_{i=1}^{n} \frac{1}{\sigma(f_i)^2} \right)^{-\frac{1}{2}} \tag{7.2}$$

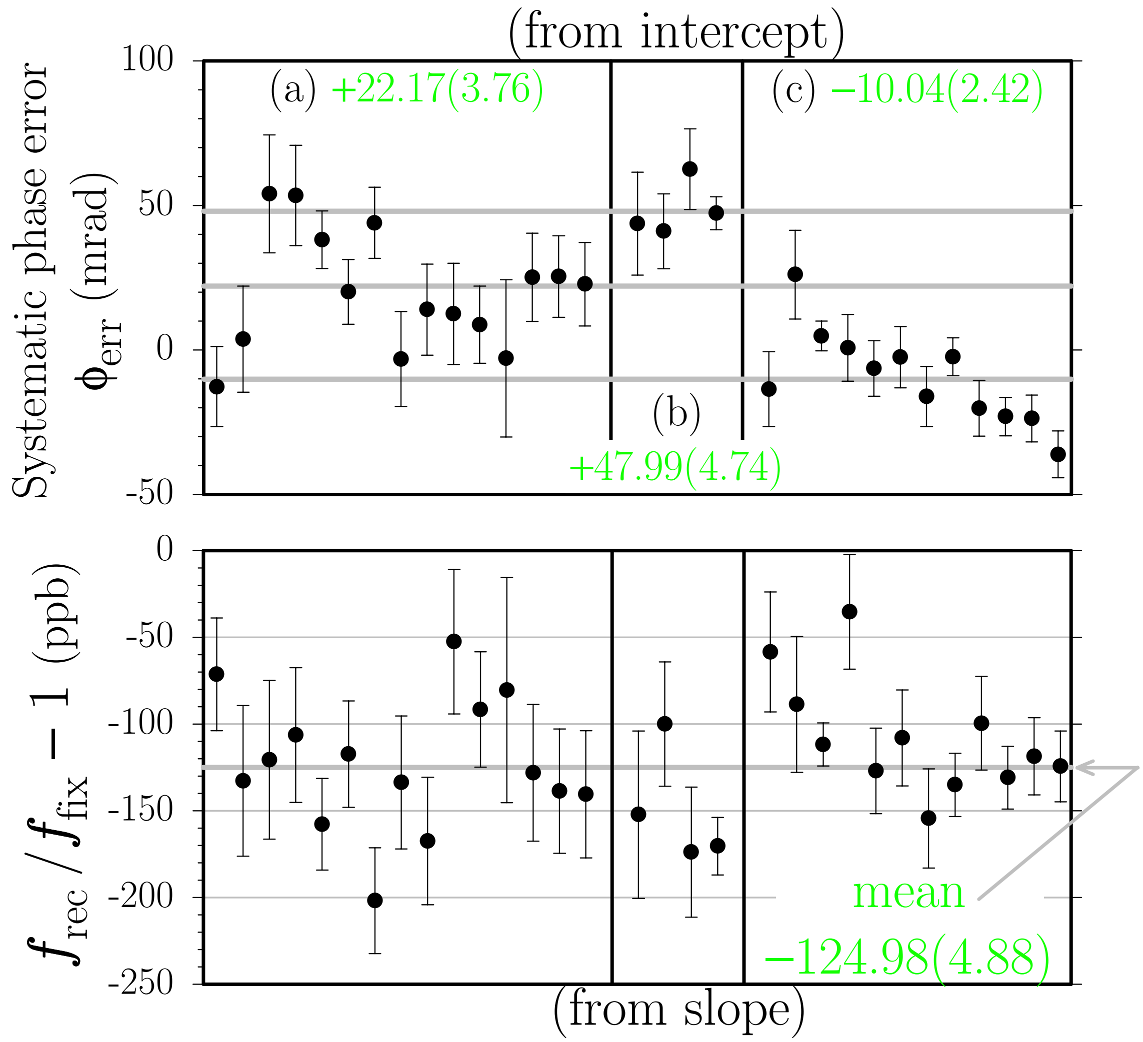


Figure 7.1: **Summary of the data taken by varying the time T between the $\pi/2$-pulses.** As described in Section 6.7, by varying the time T between the first to second and third to fourth $\pi/2$-pulses, converting the results to phase by multiplying by $4\pi(N+1)T$, and fitting with a line, we reduce each data set into two values: (upper graph) an intercept ϕ_{err} that represents the systematic phase error from the $\pi/2$-pulses and (lower graph) a slope which gives a value of the f_{rec} that is independent of this error ϕ_{err}. This data summary displayed chronologically consists in total of 2 771 repetitions and can be divided into three groups (a), (b), and (c), separated by vertical lines. We installed the common switch AOM (see Section 3.2.4) between data sets (a) and (b). We then took data with the $\pi/2$-pulses switched on and off with (c) and without (b) this common switch AOM. Comparing the mean intercepts shown in gray for (b) and (c), it is evident that although there still seems to be a remaining phase error, using the common switch AOM clearly reduces its size. We use the intercept values in each group to correct the data we took when we were not varying T. Another value for f_{rec} can be derived from the slope data in the lower graph. The fact that the slopes from different data groups are all internally consistent within their respective uncertainties indicates that this "ϕ versus T" method does successfully remove the systematic errors from the $\pi/2$-pulses.

which will be proportional to $\sigma/\sqrt{n}$ for equally weighted data $\sigma(f_i) = \sigma$. For this data set the uncertainty of the mean is 1.9 ppb. We can also compute the weighted standard deviation

$$\sigma(f_{\rm rec}) \;=\; \sqrt{\frac{\sum_{i=1}^{n} \frac{(f_i - \langle f_{\rm rec}\rangle)^2}{\sigma(f_i)^2}}{\sum_{i=1}^{n} \frac{1}{\sigma(f_i)^2}}} \tag{7.3}$$

Unfortunately, these data must be corrected for the systematic phase shift from the $\pi/2$-pulses discussed in Section 6.7. As part of identifying and fixing this phase error, we took data varying the free evolution time T on a number of occasions. We fit a line to each data subset where we varied T. As discussed in Section 6.7, the slope of the line is a measurement of the recoil frequency and the intercept is the systematic phase shift from the light pulses. By averaging the intercept values from these data shown in upper graph of Figure 7.1, we can determine a mean intercept $\langle\phi_{\rm err}\rangle$. By rewriting equation (6.50) we can use this average intercept to correct all of our measurements by an amount

$$\Delta f_{\rm rec} = \frac{-\langle\phi_{\rm err}\rangle}{4\pi(N+1)T} \tag{7.4}$$

Since the size of this correction depends on the particular values of T and N, we have to correct each individual data point before averaging all of the data together. Associated with this mean intercept is the statistical uncertainty of the mean $\sigma(\langle\phi_{\rm err}\rangle)$ and the weighted standard deviation (defined in equation (7.3) for $f_{\rm rec}$) $\sigma(\phi_{\rm err})$, which is equivalent to an uncertainty in $f_{\rm rec}$ of

$$\frac{\sigma(\phi_{\rm err})}{4\pi(N+1)T}$$

Thus, the corrected data set now looks like

$$\big\{f_i, \sigma(f_i)\big\} \;\rightarrow\; \left\{f_i - \frac{\langle\phi_{\rm err}\rangle}{4\pi(N+1)T},\; \sqrt{\sigma(f_i)^2 + \left(\frac{\sigma(\phi_{\rm err})}{4\pi(N+1)T}\right)^2}\right\} \tag{7.5}$$

which can be averaged with the new uncertainty values as weighting factors to give a corrected weighted mean of

$$f_{\mathrm{rec}}^{(\mathrm{corrected})} = 15\,006.277\,0653\,(627)\,(407)\,(477)\,\mathrm{Hz} \tag{7.6}$$

where the numbers in parentheses () are the total uncertainty of the mean, the contribution from the statistical, and the systematic uncertainties, equivalent to a fractional uncertainty of 4.2, 2.7, and 3.2 ppb, respectively. The accuracy of this value requires that the intercept values shown in the top graph of Figure 7.1 be constant. If these values for ϕ_{err} drift from data run to data run, the mean correction $\langle\phi_{\mathrm{err}}\rangle$ may not accurately represent the phase error from the $\pi/2$-pulses for each data set. One possible source of drift is the position of the Raman beams relative to the AOM crystals. Due to inhomogeneities of the sounds waves propagating through the AOM crystals, the shape and timing of the off-to-on and on-to-off edges of the light pulses depend on exactly where the beams pass through the individual AOM crystals. If from run to run the Raman beams move relative to the AOM crystals, the exact pulse shape and thus the net phase error ϕ_{err} may change.

Another value for the recoil frequency that is much less sensitive to such long term drifts can be computed from the subset of data shown in lower graph of Figure 7.1. These data which represent around 2 800 measurements of all four interferometer geometries are the fitted slopes from each of the data subsets for which we varied the free evolution time T. From each slope value, we compute f_{rec} according to

$$f_{\mathrm{rec}} - f_{\mathrm{fix}} = \frac{(\mathrm{slope})}{4\pi(N+1)} \tag{7.7}$$

The mean of all of these converted slope values gives

$$\boxed{f_{\mathrm{rec}}^{(\mathrm{slope})} = 15\,006.276\,9996\,(874)\,(732)\,(477)\ \mathrm{Hz}} \tag{7.8}$$

with the uncertainties equivalent to a fractional error of 5.8 ppb total uncertainty, 4.9 ppb statistical, and 3.2 ppb systematic uncertainty. Although this value is roughly 30% statistically less certain than $f_{\mathrm{rec}}^{(\mathrm{corrected})}$ in equation (7.6), $f_{\mathrm{rec}}^{(\mathrm{slope})}$ will be our final value. Since it was measured simultaneously with $\phi_{\rm err}$, the correction for phase errors of the $\pi/2$-pulses is more reliable.

7.2 Determining α

Table 7.1: **Current values** used to calculate the fine structure constant α from our measurement of $f_{\rm rec} = h/m_{\rm Cs}(1/\lambda_{\rm eff}^2)$, where $1/\lambda_{\rm eff} = [f(F{=}3 \to 3') + f(F{=}4 \to 3')]/c$ is the effective two-photon inverse wavelength of the two $6S_{1/2}$ to $6P_{1/2}(F{=}3')$ transitions. For comparison, the last row gives the current accepted value for α^{-1}

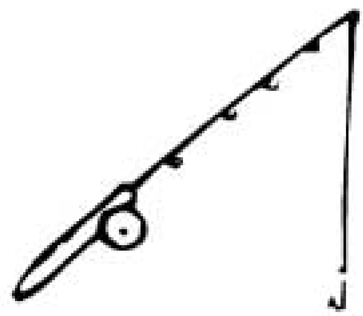
Quantity	Value	Precision	Source
Rydberg constant, R_∞	10 973 731.568 549(83) $\rm m^{-1}$	0.0076 ppb	[3]
Cesium transitions: $f(F{=}3 \to 3')$ $f(F{=}4 \to 3')$	 335 120 562 838(43) kHz 335 111 370 206(43) kHz	0.13 ppb	[63]
Cesium mass, $m_{\rm Cs}$	132.905 451 931(27) amu	0.20 ppb	[64]
Proton mass, $m_{\rm p}$	1.007 276 466 88(13) amu	0.13 ppb	[3]
Proton/electron mass, $\dfrac{m_{\rm p}}{m_{\rm e}}$	1 836.152 6675(39)	2.1 ppb	[3]
Fine structure constant, α^{-1}	137.035 999 76(50)	3.6 ppb	[3]

Using equation (1.2) and the current values given in Table 7.1 for the Rydberg constant, the cesium and proton masses, the proton to electron mass ratio, and the frequencies of our two recoil lasers, we calculate a value for the fine structure constant from our measurement of the recoil frequency $f_{\rm rec}$. Using $f_{\rm rec}^{\rm (slope)}$ in equation (7.8),

$$\boxed{\alpha^{-1} \;=\; 137.035\,999\,710(427)(401)(148) \;=\; \alpha(\text{CODATA}) - 0.37\ \text{ppb}} \tag{7.9}$$

where the uncertainties are equivalent to a 3.1 ppb total uncertainty, a 2.9 ppb uncertainty from the uncertainty of $f_{\rm rec}^{\rm (slope)}$, and a 1.1 ppb total uncertainty from the other measurements in Table 7.1. For comparison, this value is plotted in Figure 1.1 along with the results from the other techniques for measuring α.

Chapter 8



Future prospects

The several parts per billion uncertainty of our final result is very close to the performance limit of the current version of this experiment. Statistics and systematic uncertainties contribute almost equally to the final uncertainty. Because it would require running the experiment continuously for over one month, we cannot realistically expect to improve the current statistical uncertainty by more than a factor of two.

There are several possibilities for improving the sensitivity of the measurement. One of the most direct ways is to increase the size of the recoil, with multiple photon processes, higher order scattering, or simply using a lighter atom. Other possibilities involve different interferometer geometries, particularly those that enclose much larger areas of phase space. Another completely different avenue might involve improving the signal-to-noise ratio of the current experiment by inventing a new technique to simultaneous measure the phase from both conjugate interferometers. If the same light field could simultaneously generate both interferometers, the final phase difference from simultaneous interferometers would be much less sensitive to motion of the optical wavefronts due to vibrations and electronic noise. Also, by measuring the number of atoms in both hyperfine states (instead of just one state) after the final interference point, the determination of the interferometer phase could made less sensitive to amplitude fluctuations of the final detection signal.

In addition to the measurement sensitivity, the systematic uncertainty would also have to be improved. Currently, this uncertainty is determined primarily by uncertainties in the magnetic field shifts, the Sagnac phase, the accuracy of our lock to the atomic line, and the dependence on the laser beam polarization, all of which could potentially be controlled to better than 1 ppb in a future version of this experiment. What also should be addressed is the long term drift of the systematic phase shift ϕ_{err} from the $\pi/2$-pulses. Generating the $\pi/2$-pulses with off-resonant Raman transfer instead of adiabatic passage may completely eliminate this problem. Other systematic effects which might prove more difficult to control are collisional and dispersive effects from the cold atoms, which according our theoretical estimates enter at around one ppb in f_{rec}, or 0.5 ppb in α.

Appendix A

Transition strengths

The purpose of this section is to determine the saturation intensity and thereby the Rabi frequency for the cesium transitions we use in our interferometers. We will not derive the expressions from basic principles. Instead we focus on the often more troublesome part of this exercise: correctly combining a consistent set of expressions. After deriving an expression relating the Rabi frequency to the saturation intensity for our particular choice of definitions, we will discuss the different definitions found in the literature for the electric field, Rabi frequency, and saturation intensity.

A.1 Rabi frequency

We wish to evaluate the strength of the coupling when an electric dipole allowed transition between an atom's internal electronic states $|1\rangle$ and $|2\rangle$ is driven by an electric field

$$\mathbf{E}(t) = \mathbf{E}\cos(\omega t) \tag{A.1}$$

In this external electric field, the atomic dipole $\mathbf{d}$ has energy $-\mathbf{d}\cdot\mathbf{E}$, so the transition strength is

$$\rho_{12}^2 = |\langle 1|\mathbf{d}\cdot\mathbf{E}|2\rangle|^2 \tag{A.2}$$

with which is associated the Rabi frequency

$$\Omega = \frac{\mathbf{d} \cdot \mathbf{E}}{\hbar} \tag{A.3}$$

This transition matrix element in equation (A.2) can subdivided into radial and angular parts. The angular part describes the coupling between different angular momentum states completely independent of the particular atomic species. However, for every species there will always be one particular excited state to which only one other angular momentum state is coupled. For this strongest "swing transition", the angular part of equation (A.2) will be unity and the radial part can be written simply as

$$\rho_{12}^2 = d^2 E^2 \tag{A.4}$$

where size of the electric dipole can be written [65]

$$d^2 = \frac{3\hbar}{4k^3}\Gamma \tag{A.5}$$

where $\lambda = (2\pi)/k$ is the transition wavelength and $\tau = 1/\Gamma$ is the excited state lifetime. The saturation intensity I_{sat}, defined as the electric field intensity when one quarter of the population is in the excited state, can also be written in terms of k and Γ [65]

$$I_{\text{sat}} = \frac{\hbar c k^3 \Gamma}{12\pi} \tag{A.6}$$

For our definition of the electric field, in MKS units the field intensity is

$$I = \frac{c}{8\pi} E^2 \tag{A.7}$$

which can be written in terms of the Rabi frequency using equation (A.3)

$$I = \frac{c}{8\pi} \frac{\hbar^2 \Omega^2}{d^2} \tag{A.8}$$

Inserting expression (A.5) for the magnitude of the electric dipole, we have

$$I = \frac{c}{8\pi}\hbar^2\Omega^2\frac{4k^3}{3\hbar}\frac{1}{\Gamma} \tag{A.9}$$

which finally can be expressed in terms of the saturation intensity

$$\boxed{\frac{I}{I_{\text{sat}}} = 2\,\frac{\Omega^2}{\Gamma^2}} \tag{A.10}$$

A.1.1 Alternate definitions

We now list alternative ways of defining the electric field, Rabi frequency, and saturation intensity and summarize them in Table A.1.

Electric field

The electric field is often defined as

$$\mathbf{E}(t) = \mathbf{E}' e^{i\omega t} \ + \ \text{complex conjugate} \tag{A.11}$$

In order for this expression to be equivalent to relation (A.1), we must have $E' = E/2$

Rabi frequency

To simplify the expression for the Hamiltonian, the Rabi frequency is often written as

$$\Omega' = \frac{\mathbf{d}\cdot\mathbf{E}}{2\hbar} \tag{A.12}$$

to represent the oscillation of the probability amplitude instead of the population. Comparing this definition with equation (A.3), we have $\Omega' = \Omega/2$.

Saturation intensity

And finally, the saturation intensity is sometimes defined as twice the value given in equation (A.6): $I'_{\text{sat}} = 2I_{\text{sat}}$

Electric Field $E = 2E'$	Rabi frequency $\Omega = 2\Omega'$	Saturation intensity $I_{\text{sat}} = \frac{1}{2} I'_{\text{sat}}$	$\frac{I}{I_{\text{sat}}} = (\text{constant}) \frac{\Omega^2}{\Gamma^2}$ (constant)
Here	Here	Here	2
Here	Here	Elsewhere	1
Here	Elsewhere	Here	8
Here	Elsewhere	Elsewhere	4
Elsewhere	Here	Here	8
Elsewhere	Here	Elsewhere	4
Elsewhere	Elsewhere	Here	32
Elsewhere	Elsewhere	Elsewhere	16

Table A.1: **Different conventions relating the Rabi frequency to the saturation intensity.** The Rabi frequency Ω is related to the intensity of the driving field though $I/I_{\text{sat}} = (\text{constant})\Omega^2/\Gamma^2$, where (constant) is a numerical factor whose value depends on the particular definitions of the electric field, Rabi frequency, and saturation intensity. For the conventions given in equations (A.1), (A.3), and (A.6) that we adopt, this constant is 2, as in equation (A.10). This table shows the different values for this constant depending on the particular convention begin used.

A.1.2 Cesium

We now calculate the saturation intensity and Rabi frequency for the particular cesium transitions we use for the interferometers. We start with the measured saturation intensity for the cesium D2 $F{=}4, m_F{=}4 \rightarrow F{=}5, m_F{=}5$ closed transition at 853.356 nm

$$I_{\text{sat}}(D2) = 1.12\,\frac{\text{mW}}{\text{cm}^2} \tag{A.13}$$

From equation (A.6), the saturation intensity scales as $I_{\text{sat}} \propto \Gamma\lambda^{-3}$. The lifetimes of the D1 and D2 excited states are 34.8 and 30.4 ns [66], respectively, so for the D1 transitions at $\sim$894.6 nm

$$\begin{aligned} I_{\text{sat}}(\text{D1}) &= I_{\text{sat}}(\text{D2}) \left(\frac{\tau_{\text{D1}}}{\tau_{\text{D2}}}\right)^{-1} \left(\frac{\lambda_{\text{D1}}}{\lambda_{\text{D2}}}\right)^{-3} \\ &= 1.12\,\frac{\text{mW}}{\text{cm}^2} \left(\frac{34.8}{30.4}\right)^{-1} \left(\frac{894.6}{852.356}\right)^{-3} = 0.846\,\frac{\text{mW}}{\text{cm}^2} \end{aligned} \tag{A.14}$$

Since the saturation intensity also scales inversely with ρ_{12}^2, we must also evaluate the angular terms for the particular angular momentum states we are addressing. In the weak magnetic field limit, the angular momentum can be reduced to a total angular momentum quantum number F and its component m_F along the quantization axis defined by the direction of the external magnetic field. The angular part is given by

$$\mu^2 = \left|\left\langle F_1, m_{F1} \middle| C_{+1}^{(1)} \middle| F_2, m_{F2} \right\rangle\right|^2 \tag{A.15}$$

where $C_p^{(\Delta L)}$ is the rank $\Delta L\!=\!1$ Racah tensor with $p = -1, 0, +1$ for $\hat{\sigma}_-$, $\hat{\pi}$, and $\hat{\sigma}_+$ polarized light, respectively. In particular, for photons ($\Delta L\!=\!1$),

$$C_0^{(1)} = \cos(\theta) \tag{A.16}$$

$$C_{\pm 1}^{(1)} = \mp\frac{1}{\sqrt{2}}\sin(\theta)e^{\pm i\phi} \tag{A.17}$$

where θ and ϕ are the angular coordinates for the radial vector. The cesium $6S_{1/2}$ ground state has spin quantum numbers $L = 0$, $S = \frac{1}{2}$, and $J = L + S = \frac{1}{2}$. The cesium nucleus has spin quantum number $I = \frac{7}{2}$, so the total spin quantum number for this state is $F = J + I = 3$ or 4, and similarly for the $6P_{1/2}$ D1 excited state, except $L = 1$. From repeated applications of the Wigner-Eckhart theorem

$$\begin{aligned}
\left|\left\langle F_1, m_{F1} \middle| C_{+1}^{(1)} \middle| F_2, m_{F2} \right\rangle\right|^2 =& \\
(-1)^{F-m_f} &\begin{pmatrix} F' & 1 & F \\ -m_{F^0} & q & m_F \end{pmatrix} \\
&\delta_{II^0}\sqrt{2F+1}\,\sqrt{2F'+1}\,(-1)^{J+I+F^0+1}\begin{Bmatrix} I & J & F \\ 1 & F' & J' \end{Bmatrix} \\
&\delta_{SS^0}\sqrt{2J+1}\,\sqrt{2J'+1}\,(-1)^{L+S+J^0+1}\begin{Bmatrix} S & L & J \\ 1 & J' & L' \end{Bmatrix} \\
&(-1)^{L_> - L}\sqrt{L_>}
\end{aligned} \tag{A.18}$$

where () and {} are the Wigner 3-j and 6-j symbols, respectively, and $L_>$ is the larger of L and $L' = L \pm 1$ [67]. Expression A.18 is evaluated for all of the allowed

D1 transitions in Table A.2.

		$F_1 \to F_2$							
m_{F1}	m_{F2}	$3 \to 3$		$3 \to 4$		$4 \to 3$		$4 \to 4$	
−4	−4							1/9	16
−4	−3					7/36	28	1/36	4
−3	−4			7/36	28			1/36	4
−3	−3	1/16	9	7/144	7	7/144	7	1/16	9
−3	−2	1/48	3	1/144	1	7/48	21	7/144	7
−2	−3	1/48	3	7/48	21	1/144	1	7/144	7
−2	−2	1/36	4	1/12	12	1/12	12	1/36	4
−2	−1	5/144	5	1/48	3	5/48	15	1/16	9
−1	−2	5/144	5	5/48	15	1/48	3	1/16	9
−1	−1	1/144	1	5/48	15	5/48	15	1/144	1
−1	0	1/24	6	1/24	6	5/72	10	5/72	10
0	−1	1/24	6	5/72	10	1/24	6	5/72	10
0	0			1/9	16	1/9	16		
0	+1	1/24	6	5/72	10	1/24	6	5/72	10
+1	0	1/24	6	1/24	6	5/72	10	5/72	10
+1	+1	1/144	1	5/48	15	5/48	15	1/144	1
+1	+2	5/144	5	5/48	15	1/48	3	1/16	9
+2	+1	5/144	5	1/48	3	5/48	15	1/16	9
+2	+2	1/36	4	1/12	12	1/12	12	1/36	4
+2	+3	1/48	3	7/48	21	1/144	1	7/144	7
+3	+2	1/48	3	1/144	1	7/48	21	7/144	7
+3	+3	1/16	9	7/144	7	7/144	7	1/16	9
+3	+4			7/36	28			1/36	4
+4	+3					7/36	28	1/36	4
+4	+4							1/9	16

Table A.2: **The angular matrix elements:** $\left|\langle F_1, m_{F1}|\, C_p^{(1)} |F_2, m_{F2}\rangle\right|^2$ evaluated for the cesium D1 transition, where $C_p^{(1)}$ is the rank 1 Racah tensor with $p = -1, 0, +1$ for $\hat{\sigma}_-$, $\hat{\pi}$, or $\hat{\sigma}_+$ polarized light. Cesium has nuclear spin quantum number $I = \frac{7}{2}$, so the possible values for $F = I + J$ are 3 and 4. Both values in each column give the value of the matrix element, but the second value is normalized to the smallest matrix element in the entire manifold: 1/144.

For the D1 transition there is no swing state[1] that couples to only one ground state. Thus, the saturation intensity value from equation (A.14) corresponds to the total coupling out of the excited state. The $F=4', m_F=+4'$ state, for instance, couples to the $F=4, m_F=+4$, $F=4, m_F=+3$, and $F=3, m_F=+3$ states with strengths, 16, 4, and 28, respectively, for a total of $(16+4+28)/144 = 1/3$. This sum over all possible angular momentum couplings is the same for all excited states. Now for our transitions between the $F=4, m_F=0$ and $F=3, m_F=0$ ground states and the $F=3', m_F=+1'$ excited states, the angular matrix elements both happen to be $1/24$, so the saturation intensity for these transitions is

$$\begin{aligned} & I_{\text{sat}}(\text{D1}:|3,0\rangle \rightarrow |3',+1'\rangle) = I_{\text{sat}}(\text{D1}:|4,0\rangle \rightarrow |3',+1'\rangle) \\ &= I_{\text{sat}}(\text{D1}) \left(\frac{\mu^2(|3,0\rangle \rightarrow |3',+1'\rangle)}{\mu^2(|4',4'\rangle \rightarrow |4,4\rangle) + \mu^2(|4',4'\rangle \rightarrow |4,3\rangle) + \mu^2(|4',4'\rangle \rightarrow |3,3\rangle)} \right)^{-1} \\ &= I_{\text{sat}}(\text{D1}) \left(\frac{6}{16+4+28} \right)^{-1} = \left(0.846\,\frac{\text{mW}}{\text{cm}^2} \right) \left(\frac{6}{48} \right)^{-1} = 6.768\,\frac{\text{mW}}{\text{cm}^2} \end{aligned} \tag{A.19}$$

and similarly for the $F=4', m_F=+1'$ excited state

$$\begin{aligned} & I_{\text{sat}}(\text{D1}:|3,0\rangle \rightarrow |4',+1'\rangle) = I_{\text{sat}}(\text{D1}:|4,0\rangle \rightarrow |4',+1'\rangle) \\ &= I_{\text{sat}}(\text{D1}) \left(\frac{\mu^2(|3,0\rangle \rightarrow |4',+1'\rangle)}{\mu^2(|4',4'\rangle \rightarrow |4,4\rangle) + \mu^2(|4',4'\rangle \rightarrow |4,3\rangle) + \mu^2(|4',4'\rangle \rightarrow |3,3\rangle)} \right)^{-1} \\ &= I_{\text{sat}}(\text{D1}) \left(\frac{10}{16+4+28} \right)^{-1} = \left(0.846\,\frac{\text{mW}}{\text{cm}^2} \right) \left(\frac{10}{48} \right)^{-1} = 4.061\,\frac{\text{mW}}{\text{cm}^2} \end{aligned} \tag{A.20}$$

Our Raman lasers typically have a power of $P \sim 12$ mW at the atoms. With a Gaussian waist radius of $w_0 \simeq 0.96$ cm, the peak intensity is

$$I = \frac{P}{\pi w_0^2} = \frac{12\,\text{mW}}{\pi(0.96\,\text{cm}^2)} = 4.14\,\text{mW/cm}^2 \tag{A.21}$$

which gives saturation parameters $s = I/I_{\text{sat}} = 0.612$ and 1.02 for the $F=3'$ and $F=4'$ excited states, respectively. Using equation A.10, the typical Rabi frequencies

[1] For D2, the excited states $F=5', m_F=\pm 5'$ are called "swing" states because they couple to only the $F=4, m_F=\pm 4$ ground states.

for our lasers are therefore

$$\Omega(F=3') = \Gamma\sqrt{\frac{I}{2I_{\text{sat}}}} = 4.573\,\text{MHz}\sqrt{\frac{0.612}{2}} = 2.53\,\text{MHz} \tag{A.22}$$

$$\Omega(F=4') = \Gamma\sqrt{\frac{I}{2I_{\text{sat}}}} = 4.573\,\text{MHz}\sqrt{\frac{1.02}{2}} = 3.27\,\text{MHz} \tag{A.23}$$

A.2 Photon-cesium cross-section

Using the angular matrix elements presented in Table A.2 and those for the D2 transition, we can calculate photon-cesium scattering cross-sections using

$$\sigma_{12} = \frac{3\lambda_{12}^2}{2\pi}\frac{3}{2F_1+1}\sum_{m_1=-F_1}^{+F_1}\sum_{m_2=-F_2}^{+F_2}\left|\left\langle F_1, m_1\,\middle|\,C_{+1}^{(1)}\,\middle|\,F_2, m_2\right\rangle\right|^2 \tag{A.24}$$

due the presence of some atomic transition with wavelength λ_{12} between states $|1\rangle$ and $|2\rangle$. In this equation we assume the atom is equally likely to be in any of the magnetic sublevels and without loss of generality, that the incident light is $\hat{\sigma}_+$ polarized. The cross-sections for the cesium transitions are summarized in Table A.3

F_1	F_2	λ_{12} (nm)	$\frac{3\lambda_{12}^2}{2\pi}$ (cm^2)	$2F_1{+}1$	$\sum\sum$	σ_{12} (cm^2)
3	3	894.581	3.821×10^{-9}	7	7/36	1.838×10^{-10}
3	4	894.578	3.821×10^{-9}	7	7/12	5.515×10^{-10}
4	3	894.606	3.821×10^{-9}	8	14/27	4.290×10^{-10}
4	4	894.603	3.821×10^{-9}	8	10/27	3.064×10^{-10}
3	2	852.336	3.469×10^{-9}	7	5/9	4.769×10^{-10}
3	3	852.335	3.469×10^{-9}	7	7/12	5.007×10^{-10}
3	4	852.335	3.469×10^{-9}	7	5/12	3.576×10^{-10}
4	3	852.357	3.469×10^{-9}	8	14/81	1.298×10^{-10}
4	4	852.357	3.469×10^{-9}	8	14/27	3.894×10^{-10}
4	5	852.356	3.469×10^{-9}	8	88/81	8.160×10^{-10}

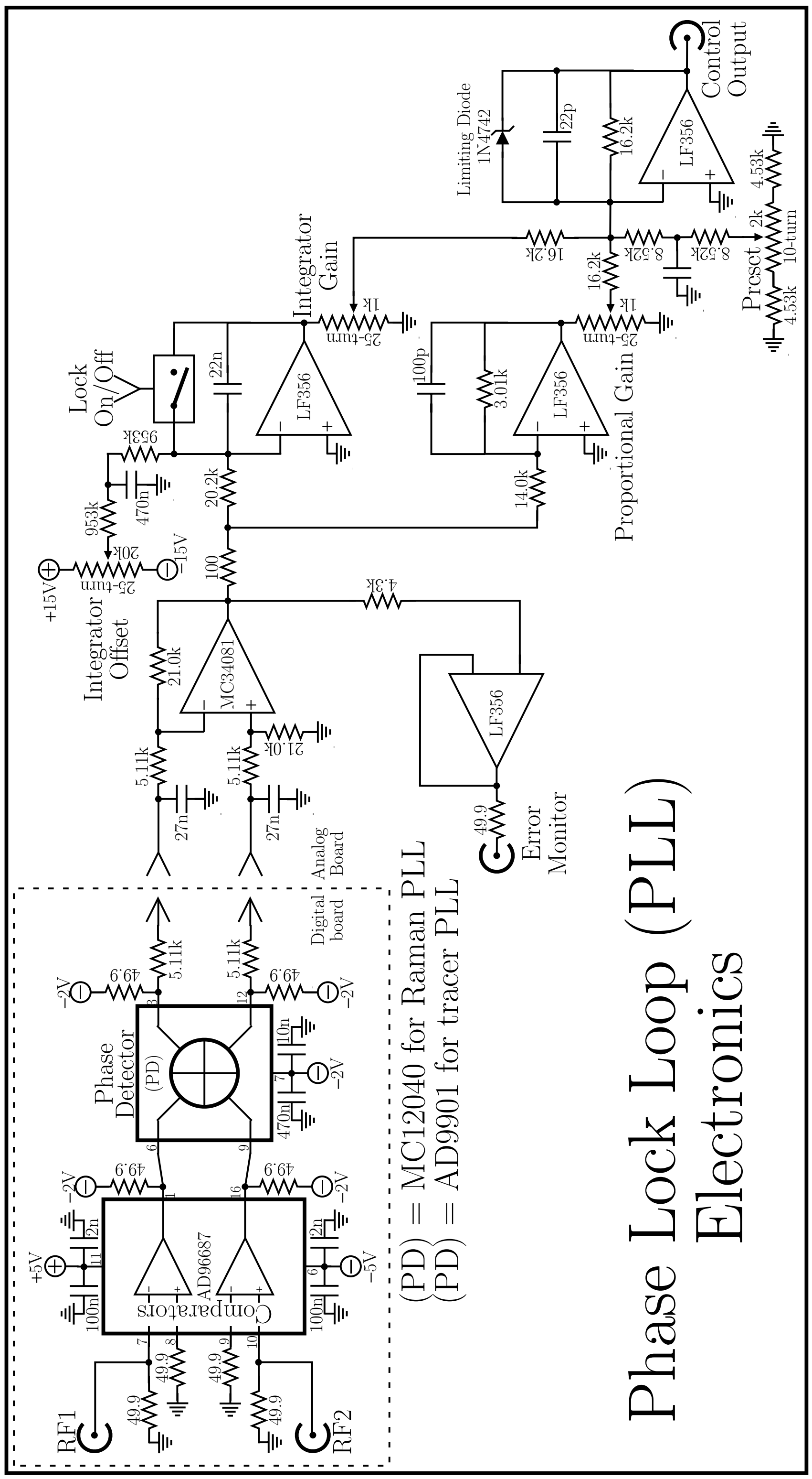
Table A.3: **Photon-atom scattering cross-sections** σ_{12} for each of the cesium transitions with wavelength λ_{12} between states with total angular momentum quantum numbers F_1 and F_2. The double sum ($\sum\sum$) indicates the sum over magnetic sublevels as in equation (A.24).

Appendix B

Phase lock loop electronics

The electronics for the Raman and tracer phase lock loops (PLLs) are designed to detect the phase difference between two sinusoidal radio frequency (rf) inputs and produce a control voltage proportional to this difference. For the tracer PLL, this output signal sets the frequency of a 100 MHz voltage controlled oscillator (VCO) that generates the rf signal for the $F=4$ switchyard AOMs (see Section 3.2.5). The Raman PLL controls a 9.3 GHz VCO that drives the 9 GHz electro-optic modulator (EOM) used to generate $F=3$ light from $F=4$ light. For each PLL, one rf inputs comes from a precision reference oscillator, while the other input comes from a photodiode detecting the beat signal between two lasers, 180 MHz for the tracer PLL and 9.280 GHz mixed down to 12.631 77 MHz for the Raman PLL.

To reduce the sensitivity to amplitude fluctuations, a comparator first converts the sinusoidal signal into an ECL square wave. A phase detector chip (MC12040 and AD9901 for the Raman and tracer PLLs, respectively) produces an ECL output signal that oscillates at the input frequency and whose duty cycle is proportional to the phase delay between the rising edges of the two input square waves. By filtering this signal with an RC filter ($\tau \simeq 140$ ns), we generate an analog error signal that we then transform using proportional and integral gains that are set to achieve optimal steady-state and dynamic performance. To prevent fast switching signals in the digital components from leaking over and disturbing the slower analog electronics, we build these two sections on separate boards, connected only by a single twisted pair.


Phase Lock Loop (PLL) Electronics
(PD) = MC12040 for Raman PLL
(PD) = AD9901 for tracer PLL
RF1
RF2
Comparators
AD96687
Phase Detector
(PD)
Digital board
Analog Board
Integrator Offset
Lock On/Off
Integrator Gain
Proportional Gain
Limiting Diode 1N4742
Control Output
Preset
Error Monitor
MC34081
LF356
25-turn
10-turn
+5V
−5V
−2V
+15V
−15V

Appendix C

Computer code

C.1 AltInt.BAS

The AltInt code written in BASIC and compiled by *Microsoft* QuickBasic 4.5 under DOS controls the entire experiment. It is separated by function into four modules. The main module, *AltInt*, calculates the atomic trajectories, computes the two-photon resonance conditions accounting for gravity and photon recoils, generates the adiabatic transfer pulse shapes for the selected interferometer(s), and programs all of the remote devices via general purpose interface bus (GPIB) connections.

```
'Program AltIntXX ---- Creates Doppler-sensitive interferometer sequences
'                     using adiabatic transfer pulses.  It allows velocity
'                     preselection with any number of complete transfer
'                     pulses with keff fixed upward.  Then it creates either
'                     two pairs of beamsplitter pulses separated by an
'                     arbitrary number of complete transfer pulses, or simply
'                     just a sequence of complete transfer pulses.  For the
'                     present vertical Raman beams, we must shift the Raman
'                     beam detuning about 21.9 kHz/ms to compensate for the
'                     frequency shift from gravity.  The final pulse can leave
'                     the signal atoms off in either the |3,0> or the |4,0>
'                     state.  Because of the large number of atoms remaining
'                     in the outer magnetic sublevels, either one or two
'                     DF Raman transitions should be used to detect only atoms
'                     in m=0.
'                        This program allows point-by-point interleaving of
'                     measurements taken for the four interferometers created
'                     by addressing different states after the first beamsplit-
'                     ter pair, and by switching the direction of keff for
'                     every pulse after the preselection pulse(s).
'                        AltInt4X programs, as AltInt30, use the two analog
'                     outputs of the Keithley PCIP-AWFG/2 arbitrary waveform
'                     generator for the F=3 and F=4 intensity controls, and
'                     four of the digital channels for the common intensity
'                     control, the strobe signal for the FSK synthesizer, and
'                     up and down control signals for the orientation of keff.
'                     The feed forward patterns are still created with an
'                     HP33120A synthesizer.  The data-collection function
'                     Scan% is contained in an independent module Data4X.BAS,
'                     and the plotting and fitting routines are in PF4X.BAS.
'                     The XMS routines are contained in XMAW.C.
'                        AltInt43 is modified from AltInt41 by creating along
'                     with the feed forward pattern a similar pattern for the
'                     tuning voltage of the double-passed AO in the Cs lock.
'                     This pattern is stored on a 2nd HP33120A synthesizer and
'                     is used to shift the one-photon detuning of the Raman
'                     beams to compensate for Doppler shifts at every pulse.
'
'                        The flow of the program is as follows:
'
'                       (1) Menu for adjusting waveform parameters
'                       (2) Simultaneously create list of pulse center times
'                              and frequencies, and generate the gate, FF,
'                              and Cs lock waveforms
'                       (3) Download gate waveform
'                       (4) Create AWFG waveforms (F=3, F=4, F=3+4, keff up,
'                              keff down, and FSK strobe)
'                       (5) Save AWFG waveform to XMS
'                       (6) Download FF and Cs lock patterns to synthesizers
'                       (7) Repeat (4)-(6) for all interferometers in sequence
'                       (8) Use Scan function to collect data
'                       (9) Complete the data file
'                      (10) Use Plot routine to plot and fit the data
'                      (11) Return to Menu
'Revision History:
'43  3/8/97    Convert the variables Tc(), dTc, g0 which are used to trace
'             the atomic trajectory (position, velocity) from single to double
'             precision.
'44  3/13/97   Use external trigger instead of line trigger for the main
'             launch time base.
'50  7/23/97   Incorporate new direct digital synthesizer (ADS-431).
'                - iMult=2 --> iMult=1     new synthesizer is not multiplied
'                - fstep=0.00532.. --> fstep=0.432
'                - reverse polarity of FSK STROBE line (AWFG:X0)
'51  7/24/97   Add another control line, AWFG:Y0, to control the RESET line
'             of the ADS-431 synthesizer to zero the phase of the frequency
'             output.
'52 10/30/97   Same as version 50, except that:
'                - ADS-431 synthesizer clocks at 1000 instead of 928 MHz
'                - FSK STROBE line back to original polarity: active LOW
'                - added variable PfPi2&(m,i) to store the frequencies of
```

```
'                  all the Pi/2-pulses so that the phase error of the ADS
'                  synthesizer can be compensated for in Function Scan
'                  of DATA??.BAS
'53 11/ 6/97   Add "autokey" feature.  Save tRam# (Raman sequence start
'             time) and ak$ (autokey string) to INI file.  tRam# is no longer
'             loaded from the SRS pulser.  It is now set before the gate
'             waveform is generated and downloaded.  tRam# is set initially
'             to the value in the INI file and from then on to the value set
'             in the menu.
'    3/ 3/98   Update the value of h/mCs using the new value of the Cesium
'             mass.
'54  3/11/98   Fix bug in my earlier patch that sends the frequencies of
'             the Pi/2 pulses to the Function SCAN(), so that the correct
'             frequencies are matched with the correct pattern.
'55  3/26/98   Add menu command 'L' so that the trigger mode (LINE,INTERNAL,
'             EXTERNAL) can be changed from the main menu.  Load the trigger
'             mode from the "ini" file.  Store the trigger mode in the "ini"
'             and output data files.
'               Move the SUB CheckName() and FUNCTION FlnmAdj() to an external
'             file fnXX.bas to leave more room for code in AltIntXX.BAS
'               Immediately after an output data file has been saved, add a
'             call to a new function FileCopy() which copies this file to an
'             alternate storage location set inside FileCopy().  In so doing,
'             we should now be able to look at the data as it comes in.
'               Start the experiment pulsing only when the patterns begin to
'             be downloaded, not when the program first starts.
'60  4/15/98   Introduce TrgMod$(0..4) which contains the text representation
'             of the trigger mode.
'               Move SUB Menu and modules specific to it to an external module
'             MENU10.BAS.
'               Improve "plot file" option from main menu.
'61  6/17/98   Add chirp control signal to AWFG:Y3.  This signal active for
'             1ms and goes inactive when the 3+4 control shuts off at the end
'             of each Pi2 pulse.  Introduce variable iTcp in clock() which
'             represents the # of additional FF points required to extend the
'             the gate signal for the Pi2 pulse out to 1ms.  Modify Pi2Pi2(),
'             PulseAS(),PulseSA().  Since the recoil direction changes between
'             Pi2-pulses 3 and 4 and I would consequently have to change the
'             chirp direction within a pattern, chirp only during Pi2-pulses
'             3 and 4.
'62  7/28/98   Add second chirp control so that we can chirp during all 4
'             Pi2-pulses.  Chirp controls are now driven by AWFG:Y2 when keff
'             UP and AWFG:Y1 when keff DOWN.
'               Fix bug in version 61 so that a T=1ms pattern can chirp.
'               Add variable kg0# for chirp rate in Hz/s.  Include kg0# in
'             output and input config files.  The DS345 can chirp for 1ms at
'             the fastest, so the chirp must start 1ms before the end of each
'             Pi/2 pulse.  This means that the frequency of the light will
'             have been shifted by the chirping shaping-AOs by an amount
'             kg0# * (1ms - TpiH).  Correct the frequencies of each Pi/2 for
'             this "chirp shift".
'               Initialize two new SRS DS345 synthesizers, mAdSnth4() for
'             chirping UP and mAdSnth5() for chirping DOWN.
'   11/ 5/98   Change linearization lookup-table filenames from aolin3/4.bin
'             to l3/4.bbn so that different tables cannot be used without
'             recompiling.
'               Add code to Pi2Pi2() so that if only the F=3 light is chirped,
'             the Cs offset lock will not be changed.
'63 11/ 6/98   Add global variable "rimb", the imbalance factor, which scales
'             the intensity of one beam for the Pi2 pulses.  PulseAS()
'             and PulseSA() now expect yA and yB instead of just y0 to
'             represent the intensity scaling factor.  The value is stored to
'             the output file and can be accessed with 's' from the menu
'64 11/24/98   Modify the chirp trigger timing so that the chirp triggers
'             occur (1ms - TdS) before the end of the Pi/2-pulses, instead of
'             just 1ms before the end.  If TdS, which nominally controls the
'             delay between pulse end and frequency strobe + beam switch, is
'             set large enough, the pulses will definitely be finished when
'             the chirping synthesizer resets from Fstop to Fstart.
'65  2/ 2/99   Modify the AltInt63's imbalance factor patch to allow the TOP
'             and BOT beam intensities to be control independently, so that
```

```
'             the overal Rabi frequency can be preserved.  Replace global
'             variable "rimb" with "rtp" and add "rbt" for the scaling factor
'             for the BOT beam intensities of the Pi/2-pulses.
'67  3/ 3/99   The chirp trigger occurs abs(dTon) too early for the 2nd and
'             4th Pi/2-pulses, because I was incorrectly determining the
'             first sample point for those pulses.  Correct Clock(), Pi2Pi2(),
'             and PulseSA().
'68  3/ 3/99   Add global variables "yTt1H" and "iTtyH" for the time spent
'             (in fraction of pulse length and in # of AWFG samples) on
'             pure-state transitions during the pi/2-pulses, so that the
'             phase-sensitive and pure-state transition times can be varied
'             independently.  Modify Clock(), PulseAS() and PulseSA().
'69  3/10/99   To fix mismatch between actual length of pi/2-pulses and
'             estimated length: redefine iTpiHC in Clock() and iC in
'             PulseSA().
'70  8/ 8/00   Change sign of Cs lock offset AO: fScCs = -25000 --> +25000
'             in CsBin$().
'71  3/ 1/01   Break CsBin$() into two functions: CsTBL%(f#) which converts
'             the frequency f# to an integer from -2047 to +2047 which
'             represents the voltage (on an 11Vpp scale not including the dc
'             offset) required to set the Cs lock to frequency f# from
'             resonance and ItoS$() which converts a 2-byte integer to a string
'             containing those two bytes.  Replace the code which slowly moves
'             the synthesizer output by downloading an 8-point arbitrary
'             wavfeform with a new subroutine SetCsOFF(f!) which changes the
'             dc offset.
'72 7/25/01     EXACTLY the same as ALTINT71.BAS, but DATA55.BAS has been
'             corrected into DATA56.BAS.
'
'*70:$INCLUDE: 'e:\lab\inc\nihpdas.inc'
'*70:$INCLUDE: 'e:\lab\inc\xmaw.inc'
'*71b
'$INCLUDE: 'inc\nihpdas.inc'
'$INCLUDE: 'inc\xmaw.inc'
'*71e

DEFINT I-N
DEFSNG A-H, O-Z

DECLARE SUB AWFG (ix, iy)
DECLARE SUB basdasg (iMode, BYVAL dummy%, nErr)
DECLARE SUB BinCon (f#, fRndB&, fRnd#)
DECLARE SUB CheckName (Flnm$, iMode)
DECLARE SUB Clock (iMode)
DECLARE SUB menu (fi$)
DECLARE SUB PiPulse (N, iFFpi, iG, iTpi)
DECLARE SUB Pi2Pi2 (iArmI)
DECLARE SUB Plot (mode, FlnmP$, psc)
DECLARE SUB PulseAB (if3(), if4(), iTt1, iTd, iTt2, Y)
DECLARE SUB PulseAS (if3(), if4(), ike)
DECLARE SUB PulseSA (if3(), if4(), ike)
DECLARE SUB SetCsOFF (f!)
DECLARE FUNCTION RdInit% (a$)
DECLARE FUNCTION Clear$ ()
DECLARE FUNCTION ItoS$ (i%)
DECLARE FUNCTION CsTBL% (f#)
DECLARE FUNCTION Fctr$ (fRnd#)
DECLARE FUNCTION FFbin$ (fRnd#)
DECLARE FUNCTION FlnmAdj$ (Flnm$, path$, ext$)
DECLARE FUNCTION FileCopy% (src$, dst$)
DECLARE FUNCTION FSKdrive% (iMode, f#, nElem&, FLckHDL&, lFSK&)
DECLARE FUNCTION iwf (x, a, if3(), Cwf())
DECLARE FUNCTION msec$ (t#)
DECLARE FUNCTION Scan% (mSt(), sh$, df$(), bl$(), pr$(), fCAve#(), fBack&(), nFSK0, iUp)
DECLARE FUNCTION Tcorr (T1, t0#, u#, du#)
DECLARE FUNCTION Tctr$ (t#)
DECLARE FUNCTION Tdet$ (t)
DECLARE FUNCTION Trnd (t#)
DECLARE FUNCTION usec$ (t#)
DECLARE FUNCTION zFmt$ (z)
DECLARE FUNCTION IncFN$ (f$)
```

```

'Constants for the cesium lock AO tuning voltage calibration table
CONST finit = "AltInt71.ini" 'File for initialization data
CONST vflnm = "e:\lab\cshp.bin"
CONST f21 = 2000# 'Scale factor from 2-photon detunings (Hz) to 1-photon (kHz)

'Miscellaneous constants
CONST fCent = 12631770# 'Offset frequency for the DF 0->0 transition
CONST iMult = 1 'Multiplication factor on output of DDS-1EB synthesizer
CONST sgndet = 1 'Sign of the detuning of the EOM from the u-wave reference
'       (This is also the sign of absolute light shift with DDS frequency.)
'CONST zpr = .086 'Vertical position (m) of probe region relative to trap
'New probe position as of 3/3/98:
CONST zpr = .081  'Vertical position (m) of probe region relative to trap
CONST yMax = 2047 'Arbitary pattern maximum
CONST fFmt = "#########.###"
CONST COM1% = &H8, COM0% = &HFFF7 'Turn ON or OFF common intensity switch
CONST BS0% = &HFFF9 'Turn OFF both beam switch bits
CONST STR1% = &H1, STR0% = &HFFFE 'Turn ON or OFF frequency strobe
CONST CHPU1% = &H4, CHPU0% = &HFFFB 'Turn ON or OFF chirp UP control
CONST CHPD1% = &H2, CHPD0% = &HFFFD 'Turn ON or OFF chirp DOWN control
CONST nAWFG0 = 1 'Extra AWFG point at start of pattern
CONST Scrn = 12 'Screen mode
CONST ipl = 0 'Print error messages?

'70:$INCLUDE: 'e:\lab\inc\AltInt.inc'
'*71:
'$INCLUDE: 'inc\AltInt.inc'

'$DYNAMIC

'free up memory for NI-DAQ DOS functions
heap.size = SETMEM(-7000)

    'Initialize Variables

    'GPIB addressing arrays
    mAdSnth1(0).primad = 27: mAdSnth1(0).secad = -1 'SRS synth (gate signal)
    mAdSnth1(1).primad = -1
    mAdSnth2(0).primad = 29: mAdSnth2(0).secad = -1 'HP synth (feed forward)
    mAdSnth2(1).primad = -1
    mAdSnth3(0).primad = 28: mAdSnth3(0).secad = -1 'HP synth (Cs detuning)
    mAdSnth3(1).primad = -1
    mAdSnth4(0).primad = 26: mAdSnth4(0).secad = -1 'SRS synth (chirp UP)
    mAdSnth4(1).primad = -1
    mAdSnth5(0).primad = 25: mAdSnth5(0).secad = -1 'SRS synth (chirp down)
    mAdSnth5(1).primad = -1
    mAdSRSP1(0).primad = 22: mAdSRSP1(0).secad = -1 'SRS pulser (DF Raman 1&2)
    mAdSRSP1(1).primad = -1
    mAdSRSP2(0).primad = 23: mAdSRSP2(0).secad = -1 'SRS pulser (shutter/blast/probe)
    mAdSRSP2(1).primad = -1
    mAdSRSP3(0).primad = 24: mAdSRSP3(0).secad = -1 'SRS pulser (Raman start)
    mAdSRSP3(1).primad = -1
    mAdAux(0).primad = 13: mAdSRSP3(0).secad = -1   'Auxillary device
    mAdAux(1).primad = -1
    mTerm(0) = -1: mTerm(1) = -1: mTerm(2) = -1 'GPIB terminator array

    iTrgMod = 4  'Experiment trigger mode: 0=internal, 1=external, 4=line
    iTM = 0      'Experiment pulsing? (0=NO, -1=YES)
    TrgMod$(0) = "INTERNAL": TrgMod$(1) = "EXTERNAL": TrgMod$(4) = "  LINE  "
    d = CHR$(13) + CHR$(10)
    nErr = 0 ' error return variable
    iAuto = 0 '0 -> return to menu after each freq scan
    'Scan modes
    Auto$(0) = "single": Auto$(1) = "auto without save": Auto$(2) = "auto save"
    Fix$(-1) = "temporally": Fix$(0) = "spatially"

    'AutoKey variables:
    ak$ = ""
    doak% = 0
    iak% = 1
```

```
'*63,*65b
    rtp! = 1!
    rbt! = 1!
'*63,*65e

    'Commands for auxillary device:
    iauxmx = 0
'    AuxS$(1) = ";DT2,1," + "0.5927" + ";"
'    AuxS$(2) = ";DT2,1," + "0.7451" + ";"
'    AuxS$(3) = ";DT2,1," + "0.8974" + ";"
'    AuxS$(4) = ";DT2,1," + "0.1357" + ";"
'    AuxS$(5) = ";DT2,1," + "0.2880" + ";"
'    AuxS$(6) = ";DT2,1," + "0.4404" + ";"
'    AuxS$(1) = ";SPFR " + "20000000.0000" + ";"
'    AuxS$(2) = ";SPFR " + "20005476.9758" + ";"
'    AuxS$(3) = ";SPFR " + "19994523.0242" + ";"
'    AuxS$(4) = ";SPFR " + "20010953.9516" + ";"
'    AuxS$(5) = ";SPFR " + "19989046.0484" + ";"
    AuxS$(1) = ";OFFS  .15"
    iaux = 1

    SCREEN Scrn: CLS : COLOR 15 'Bright white

    PRINT "Resetting the GPIB driver."
    CALL kreset(0, nErr) 'Reset the driver
    IF nErr THEN PRINT "kreset: Error #"; nErr: iQuit = -1
    PRINT "Setting GPIB timeout = 10 seconds."
    CALL kto(0, nErr, 180) 'Set time out to 10 seconds
    IF nErr THEN PRINT "kto: Error #"; nErr: iQuit = -1

    PRINT "Initializing the A/D board."
    'Initialize DAS-16 using mode 0
    iParlst(0) = &H300: iParlst(1) = 7: iParlst(2) = 1: iMode = 0: nErr = 0
    'DAS I/O Address;Interrupt level;D.M.A. level;initialize mode;error
    CALL basdasg(iMode, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 0: Error #"; nErr: iQuit = -1

    'Set DAS-16 programmable timer to 50000 Hz (assuming 10MHz jumper)
    iParlst(0) = 2: iParlst(1) = 100: iMode = 17
    'product = 200;timer set mode
    CALL basdasg(iMode, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 17: Error #"; nErr: iQuit = -1

    'Set the DAS-16 channel limits using mode 1
    iParlst(0) = 0: iParlst(1) = 0 'lower limit; upper limit
    CALL basdasg(1, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 1: Error #"; nErr: iQuit = -1

    PRINT "Initializing the digital I/O and DDS synthesizer boards."
    nErr = FSKdrive%(0, fCent / iMult, nElem&, FLckHDL&, lFSK&)
    IF nErr THEN PRINT "FSKdrive mode 0: Error #"; nErr: iQuit = -1

    PRINT "Initializing XMS driver for AWFG patterns"
    nErr = XMSInit
    IF nErr THEN PRINT "XMSInit: Error #"; nErr: iQuit = -1

    PRINT "Loading initial configuration from '"; finit'"'."
    a$ = pinit + finit
    IF RdInit%(a$) THEN SYSTEM
    'Find first available file name
    sDate$ = LEFT$(DATE$, 6) + RIGHT$(DATE$, 2)
    Flnm$ = "SC" + LEFT$(sDate$, 2) + MID$(sDate$, 4, 2) + "00.DAT"
    PRINT "Checking the filename."
    CALL CheckName(Flnm$, 0)
    FilePrfx$ = LEFT$(Flnm$, 2)

    PRINT "Calculating waveform parameters."
    CALL Clock(0) 'Determine waveform parameters

    PRINT "Loading the Cs lock AO tuning voltage calibration data."
    DEF SEG = VARSEG(iCs(0, 0))
```

```
    BLOAD vflnm, VARPTR(iCs(0, 0))
    DEF SEG

'*70b
'    PRINT "Setting the Cs lock offset to"; fOffCs; "kHz."
'    'Program an 8 point pattern to VOLATILE memory to set the output level
'    'Set Trigger Source = Bus for triggering via software control
'    outexpr = HPInit$ + CsData$
'    c$ = Csbin$(0#) 'New Cs lock setting
'    FOR N = 1 TO 8 'Generate waveform description
'      outexpr = outexpr + c$
'    NEXT N
'    'Select volatile arbitrary waveform for fast control of DC output level
'    'Trigger pattern and tell HP to wait for completion of trigger
'    outexpr = outexpr + CsSel$ + Trig$
'    'Download waveform for this output level
'    l& = LEN(outexpr)
'    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
'    IF nErr THEN PRINT "Cs synth: Error #"; nErr: iQuit = -1
'*70e
'*71b
    'Initialize Cs lock synthesizer:
    outexpr = HPInit$ + ":VOLT:OFFS?"
    'Save Settings to #3, Recall Settings from #1
    'Set Trigger Source = Bus for triggering via software control
    'Set byte order for binary transfer to NORMAL
    'Query the dc offset value
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
    IF nErr THEN PRINT "Cs synth: Error #"; nErr: iQuit = -1
    inexpr = STRING$(30, 32)
    CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSnth3(0), mTerm(0))
    IF nErr THEN PRINT "Read Cs offset Error #"; nErr: iQuit = -1
    vCsOFF0 = VAL(inexpr)       'this prorgam assumes that this dc offset
                                ' value is the voltage required to put the
                                ' Cs lock on resonance
    PRINT "Setting the Cs lock offset to"; fOffCs; "kHz."
    CALL SetCsOFF(fOffCs)
'*71e

    'Initialize Chirp synthesizers:
    outexpr = ";*SAV 9;*RCL 1;": l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth4(0), mTerm(0))
    IF nErr THEN PRINT "Chirp UP device: Error #"; nErr: iQuit = -1
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth5(0), mTerm(0))
'    IF nErr THEN PRINT "Chirp DOWN device: Error #"; nErr: iQuit = -1

    'Initialize Auxillary device(s):
'    outexpr = ";RC7;": l& = LEN(outexpr)        'SRS Pulser: recall setting 7
'    outexpr = ";*SAV 9;*RCL 1;": l& = LEN(outexpr)
'    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdAux(0), mTerm(0))
'    outexpr = ";FUNC 0;AMPL 0 VP;FREQ 0.001;": l& = LEN(outexpr)
    'set output to 0 Vpp sine wave with freq 0.001 Hz
'    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdAux(0), mTerm(0))
'    l& = LEN(AuxS$(1))
    'set offset to 0.15 Volts = 0.3 V for HighZ
'    CALL koutputStr(0, nErr, SSEGADD&(AuxS$(1)), l&, 0, mAdAux(0), mTerm(0))
'    IF nErr THEN PRINT "Aux. Device: Error #"; nErr: iQuit = -1

50  'Restore DF Raman pulse times for SRS pulser#2
    outexpr = ";ST9;RC1;": l& = LEN(outexpr) 'Recall 0-pulse DF Raman time
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
    IF nErr THEN PRINT "DF Raman timing: IEEE Error #"; nErr: iQuit = -1

    'Restore shutter, blasting, and probe times for SRS pulser#3
    outexpr = ";ST9;RC1;": l& = LEN(outexpr) 'Recall 0-pulse times
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP2(0), mTerm(0))
    IF nErr THEN PRINT "Probe timing: Error #"; nErr: iQuit = -1

    'Read in the Raman sequence start time (needed for Menu)
'    outexpr = "DT2": l& = LEN(outexpr) 'Request Raman sequence trigger time
```

```
'    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
'    IF nErr THEN PRINT "Read trigger timing request: Error #"; nErr: iQuit = -1
'    inexpr = STRING$(30, 32) 'Read in trigger time
'    CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP3(0), mTerm(0))
'    IF nErr THEN PRINT "Read trigger timing: Error #"; nErr: iQuit = -1
'    tRam# = VAL(RIGHT$(inexpr, LEN(inexpr) - 2)) 'Raman trigger time in sec.

    IF iQuit THEN 'I/O errors occurred
      PRINT "Press any key to continue, Q to return to system."
      DO: a$ = INKEY$: LOOP WHILE a$ = ""
      IF UCASE$(a$) = "Q" THEN nErr = XMSFree(-1): SYSTEM
      iQuit = 0
    END IF

    COLOR 15 'Bright white for menu
'    IF iAuto = 0 THEN CALL Menu(finit)          'Menu for scan parameters
    CALL menu(finit)                              'Menu for scan parameters

    'The frequencies are always recalculated in case tRam# or Tpr changes
    fList$(0) = "   Pulse     Tc(ms)      Freq.(Hz)     Freq.(Hz)     Freq.(Hz)     Freq.(Hz)  "
    CLS : PRINT fList$(0) 'Prepare for list of frequencies

    'Determine which sequences require an amplitude pattern to be created
    iArb = 0 'Need at least one pattern?
    FOR j = 1 TO nSeq
      m = mSeq(j)
      IF NOT iMem(m) THEN 'This pattern not in memory -> create and load
        iDwn(m) = -1: iArb = -1 'Set download flag
      END IF
      mSt(j) = m - (m = 0) 'FF pattern storage locations
    NEXT j
    iMod = 0: iModH = 0 'Reset modification flags after updating memory flags

    'Determine actual exit state from pulse sequence
    IF iINT(0) THEN 'Only pi pulses
      iArm = (-1) ^ (nPiP + nPi) 'Exit in |4> for even number of pulses
    ELSE 'Last pulses are beamsplitters
      iArm = iArmI 'Exit in selected state
    END IF

    'Collect information needed to determine atomic trajectories

    IF iArm = 1 THEN 'Read in the 1st DF Raman time for 0 photon recoils
      outexpr = ";DT2;": l& = LEN(outexpr) 'Request DF Raman time
      CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
      IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
      inexpr = STRING$(30, 32) 'Read in DF Raman time
      CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP1(0), mTerm(0))
      IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
      Tdf1(0) = VAL(RIGHT$(inexpr, LEN(inexpr) - 2))
    END IF

    'Read in the blasting time for 0 photon recoils
    outexpr = ";DT3;": l& = LEN(outexpr) 'Request blasting time
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP2(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    inexpr = STRING$(30, 32) 'Read in blasting time
    CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP2(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    Tbl(0) = VAL(RIGHT$(inexpr, LEN(inexpr) - 2))

    'Read in the 2nd DF Raman time for 0 photon recoils
    outexpr = ";DT5;": l& = LEN(outexpr) 'Request DF Raman time
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    inexpr = STRING$(30, 32) 'Read in DF Raman time
    CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP1(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    Tdf2(0) = VAL(RIGHT$(inexpr, LEN(inexpr) - 2))

    'Finally, read in the probe time for 0 photon recoils
```

```
    '  This, in combination with tRam# and the initial detuning give all the
    '     information necessary to calculate the probe time for any sequence
    outexpr = ";DT5;": l& = LEN(outexpr) 'Request probe time
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP2(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    inexpr = STRING$(30, 32) 'Read in probe time
    CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP2(0), mTerm(0))
    IF nErr THEN PRINT "IEEE Error #"; nErr: iQuit = -1
    Tpr(0) = VAL(RIGHT$(inexpr, LEN(inexpr) - 2))

    IF iQuit THEN 'I/O errors occurred
      PRINT "Press any key to continue, Q to return to system."
      DO: a$ = INKEY$: LOOP WHILE a$ = ""
      IF UCASE$(a$) = "Q" THEN nErr = XMSFree(-1): SYSTEM
      iQuit = 0
    END IF

    'Calculate the atomic trajectories for all pulse sequences, yielding the
    ' Raman detuning needed to compensate for gravity and recoils.
    'The trajectories through interferometers are calculated for a fictitious
    ' atom with a momentum which is the average of the two paths.  Such an
    ' atom will be spatially overlapped with the two paths at the final
    ' interferometer vertex, so as long as the recoil is handled correctly at
    ' that point, the trajectory calculation is valid.
    'The gating control is common to all interferometers, so it is generated
    ' along with the frequency sequences.

    'Initialize variables common to all four patterns
    ' Create first gate pulse for preselection strobe
    iwfG(0) = 0: iwfG(1) = 0: iwfG(2) = 1: iwfG(3) = yMax  'Gate OFF->ON
    iGi = 4 'Index for gate waveform coordinate list
    IF iGs > 1 THEN iwfG(4) = iGs: iwfG(5) = yMax: iGi = 6
    iFSK = 0 'Pointer for FSK pattern (Pulse counter)
    i0&(0) = 0 'First AWFG sample for preselection strobe
    iG = iGs + 1 'Gate sample counter
    nOffG = 1 'Counter of OFF gate samples
    iFF = iFFs 'Pointer for FF pattern
    Tc#(0) = iFFs * Tff# 'End of FF pts for preselection strobe
    nByte$ = LTRIM$(STR$(2 * nFF)) 'Number of binary bytes for FF string
    ndig$ = CHR$(48 + LEN(nByte$)) 'Number of digits in length of data
    FFdata$ = ":DATA:DAC VOLATILE, #" + ndig$ + nByte$

    'Initialize variables for the individual patterns
    m = 1 'Pointer for pattern storage
    FOR j = 0 TO 4
      IF iINT(j) THEN 'This pattern appears in sequence
         'Express all pulse times relative to the Raman trigger, tRam
         ' Before pulses, trajectories same for all sequences
         Tpr(m) = Tpr(0) - tRam#
         IF NOT iFix THEN
            IF iArm = 1 THEN Tdf1(m) = Tdf1(0) - tRam#
            Tbl(m) = Tbl(0) - tRam#: Tdf2(m) = Tdf2(0) - tRam#
         END IF
         'fOffP is the resonance frequency of an atom at the trigger time for
         '  a beam with keff directed upward.  The atoms are initially in F=4
         '  after Zeeman pumping, which gives the sign of the recoil shift.
         'Initial velocity after 1st strobe pulse
         v0#(m) = fOffP / keff + vR / 2 - g0# * Tc#(0)
         'Initial position after 1st strobe pulse
         z0(m) = zpr + ((v0#(m) - g0# * (Tpr(m) - Tc#(0))) ^ 2 - v0#(m) ^ 2) / 2 / g0#
         zmin(m) = 1: zmax(m) = 0 'Preset extreme points of trajectory
         'Clear waveform patterns
         iSt(m, 0) = 1 'Start out in |4,0> state
         ik(m, 0) = 1 '1st preselection pulse has keff up
      END IF
      m = j + 1 'For j>0, store seq #m in mem #j
    NEXT j

    'Generate description for preselection pi pulses"
    iRev = 0 'Don't switch keff sign between preselection pi's
    iExt = iFF0 'Extra delay before starting pattern
```

```
  FOR N = 1 TO nPiP
    CALL PiPulse(N, iFFpiP, iGpiP, iTpiP)
    IF CSRLIN > 26 THEN 'Refresh the screen for more frequencies
       PRINT : PRINT "Press any key to continue.";
       IF (NOT doak%) THEN DO WHILE INKEY$ = "": LOOP
       CLS : PRINT fList$(0)
    END IF
    iExt = 0 'Only delay before 1st pi pulse
  NEXT N

  'After preselection, flip keff for int's 3 and 4"
  IF iINT(3) THEN ik(3, iFSK) = -ik(3, iFSK)
  IF iINT(4) THEN ik(4, iFSK) = -ik(4, iFSK)

 'PRINT "->"; ik(1,iFSK); ik(2,iFSK); ik(3,iFSK); ik(4,iFSK)

  IF iINT(0) = 0 THEN 'Generate 1st pi/2 pair
    CALL Pi2Pi2(0) '0 -> Pick upper or lower trajectory, depending on m
    iExt = 0 'Extra delay completed
    FOR j = 1 TO nSeq         'step through all sequences
      IF (iINT(j)) THEN
        CALL BinCon(fC#(mSeq(j), 0), PfPi2&(j, 1), fRnd#)
          'save frequency of first Pi/2-pulse
        CALL BinCon(fC#(mSeq(j), 1), PfPi2&(j, 2), fRnd#)
          'save frequency of second Pi/2-pulse
      END IF
    NEXT j
  END IF

  'Generate main pi pulses"
  iRev = -1 'Alternate keff sign for main pi pulses
  FOR N = 1 TO nPi
    CALL PiPulse(N, iFFpi, iGpi, iTpi)
    IF CSRLIN > 26 THEN 'Refresh the screen for more frequencies
       PRINT : PRINT "Press any key to continue.";
       IF (NOT doak%) THEN DO WHILE INKEY$ = "": LOOP
       CLS : PRINT fList$(0)
    END IF
    iExt = 0 'Only delay before 1st pi pulse
  NEXT N

  IF NOT iINT(0) THEN 'Generate 2nd pi/2 pair
    CALL Pi2Pi2(iArmI) 'iArmI -> Exit state - independent of m
    FOR j = 1 TO 4            'step through all
      IF (iINT(j)) THEN       ' active interferometers
        CALL BinCon(fC#(j, 0), PfPi2&(j, 3), fRnd#)
          'save frequency of third Pi/2-pulse
        CALL BinCon(fC#(j, 1), PfPi2&(j, 4), fRnd#)
          'save frequency of fourth Pi/2-pulse
      END IF
    NEXT j
  END IF
  nFSK = iFSK + 1 'Total number of frequencies in list (+1 for fCent)
  IF nFSK <> nFreq THEN PRINT "WARNING! nFSK="; nFSK; "nFreq="; nFreq: DO: LOOP WHILE INKEY$ = ""
  nFSK0 = iFSK - 2 - iINT(0) 'Index for first FSK element to be scanned

  'Finish up gate control for final beam switch at end of next FF point
  ' Find gate sample for final beam switch
  iFF = iFF + 1: i0G = INT(fSampG# * iFF * Tff#) - 1
  iwfG(iGi) = iG: iwfG(iGi + 1) = 0: iGi = iGi + 2 'OFF
  IF i0G - iG > 1 THEN iwfG(iGi) = i0G - 1: iwfG(iGi + 1) = 0: iGi = iGi + 2
  iwfG(iGi) = i0G: iwfG(iGi + 1) = yMax 'ON
  iwfG(iGi + 2) = i0G + 1: iwfG(iGi + 3) = 0: iGi = iGi + 4 'OFF
  IF iGi > iGiMax THEN PRINT "Error: iGi="; iGi; "iGiMax="; iGiMax: DO: LOOP WHILE INKEY$ = ""

  'Prepare for list of state selection/detection pulses
  fList$(nFSK) = "   Pulse     T (ms)        T'(ms)        T'(ms)        T'(ms)        T'(ms)  "
  IF iArm = 1 THEN 'Include 1st DF Raman pulse from 4->3
    nList = nFSK + 5
    fList$(nList - 3) = "    Tdf:"
  ELSE 'No 1st DF Raman pulse
```

```
    nList = nFSK + 4
  END IF
  fList$(nList - 2) = "    Tbl:"
  fList$(nList - 1) = "    Tdf:"
  fList$(nList) = "    Tpr:"

  'Prepare for list of atomic positions
  fList$(nList + 1) = "   Pulse     z (cm)  z (cm)  z (cm)  z (cm) "
  fList$(nList + 2) = "    Min:  "
  fList$(nList + 3) = "    Max:  "
  nListZ = nList + 3
  IF NOT iINT(0) THEN
    nListZ = nListZ + 4
    fList$(nList + 4) = "pi/2 pair:": fList$(nList + 6) = fList$(nList + 4)
    fList$(nList + 5) = STRING$(10, 32): fList$(nList + 7) = STRING$(10, 32)
  END IF

  IF iFix THEN 'Tdf and Tbl at same time for all sequences
    'Open probe shutter at end of FF pattern
    Tsh = Trnd(tRam# + nFF * Tff#)
    m = mSt(1) 'Store Tdf and Tbl in location for 1st point
    Tdf2(0) = Tdf2(0) - Tbl(0) 'Blasting -> DF#2 spacing
    IF iArm = 1 THEN
       Tbl(0) = Tbl(0) - Tdf1(0) 'DF#1 -> blasting spacing
       '1st DF pulse 2ms after probe shutter opens
       Tdf1(0) = Trnd(Tsh + .002#)
       Tbl(0) = Trnd(Tdf1(0) + Tbl(0))
       df$(m) = ";DT2,1," + STR$(Tdf1(0)) 'DF#1: A = T0 + Tdf1(0)
    ELSE
       Tbl(0) = Tsh + .002 'Blasting pulse 2ms after probe shutter opens
       df$(m) = ""
    END IF
    bl$(m) = ";DT3,1," + STR$(Tbl(0)) 'blasting: B = T0 + Tbl(0)
    Tdf2(0) = Tbl(0) + Tdf2(0) 'Add blasting -> DF#2 spacing
    df$(m) = df$(m) + ";DT5,1," + STR$(Tdf2(0)) 'DF#2: C = T0 + Tdf2(0)
  ELSE
    Tsh = Tpr(0) 'Tsh will be shifted to 2ms before the earliest pulse
  END IF

  'List unadjusted times for all state selection/detection pulses
  IF iArm = 1 THEN '1st DF Raman pulse
    fList$(nList - 3) = fList$(nList - 3) + Tdet$(Tdf1(0))
  END IF
  fList$(nList - 2) = fList$(nList - 2) + Tdet$(Tbl(0))
  fList$(nList - 1) = fList$(nList - 1) + Tdet$(Tdf2(0))
  fList$(nList) = fList$(nList) + Tdet$(Tpr(0))

  'Finish up FSK data, FF, and cesium lock patterns
  IF iFF <> nFF THEN 'Wrong number of FF points
    PRINT "Error: iFF="; iFF; ", nFF="; nFF
    PRINT "Press any key to continue, Q to return to system."
    DO: a$ = INKEY$: LOOP WHILE a$ = ""
    IF UCASE$(a$) = "Q" THEN nErr = XMSFree(-1): SYSTEM
    iAuto = 0
  END IF
  'Switch back to fCent, with the FF at zero, so cavity settles
  ' Set the cesium lock at its user-selected offset-frequency
  m = 1 'Pointer for pattern storage
  FOR j = 0 TO 4 'Step through all possible patterns
    IF iINT(j) THEN 'This pattern appears in sequence
       'Include iFFs/1 FF points at start/end at fCent
       'Find binary representation of nearest accessible frequency
       'CALL BinCon(-fCent, fRndB&, fRnd#) 'f=0 to check jitter
       CALL BinCon(0#, fRndB&, fRnd#) 'f=fCent for experiment
       'Update FF string and FSK array
       b$ = FFbin$(fRnd#) 'New FF value
       c$ = ItoS$(CsTBL%(fRnd# / f21)) 'New Cs lock setting
       FOR N = 1 TO iFFs 'Add points during preselection strobe
          FF$(m) = b$ + FF$(m)
          Cs$(m) = c$ + Cs$(m)
       NEXT N
```

```
       FF$(m) = FF$(m) + b$
       Cs$(m) = Cs$(m) + c$
       IF LEN(FF$(m)) <> 2 * nFF THEN 'Wrong length for FF$
          PRINT "Error: LEN(FF$("; m; ")="; LEN(FF$(m)); ", nFF="; nFF
          DO: LOOP WHILE INKEY$ = ""
          iAuto = 0: GOTO 70
       END IF
       FSK&(iFSK, m) = fRndB& 'fCent
       FSK&(nFreq, m) = FSK&(0, m) 'Copy preselection for rotation

       'Calculate background and average scan frequencies
       CALL BinCon(fC#(m, 0) + fBackOff, fBack&(m, 0), fRnd#)
       IF NOT iINT(0) THEN CALL BinCon(fC#(m, 1) + fBackOff, fBack&(m, 1), fRnd#)
       fCAve#(m) = (fC#(m, 0) + fC#(m, 1 + iINT(0))) / 2
         'Data will be plotted vs. average offset frequency

       'Add offset tRam# and round detection pulse times
       'Generate commands for changing pulser times and list the times
       IF NOT iFix THEN 'Change Tdf and Tbl
          Tbl(m) = Trnd(tRam# + Tbl(m))
          bl$(m) = ";DT3,1," + STR$(Tbl(m)) 'blasting: B = T0 + Tbl(m)
          fList$(nList - 2) = fList$(nList - 2) + Tdet$(Tbl(m))
          Tdf2(m) = Trnd(tRam# + Tdf2(m))
          df$(m) = ";DT5,1," + STR$(Tdf2(m)) 'DF #2: C=T0+Tdf2(m)
          fList$(nList - 1) = fList$(nList - 1) + Tdet$(Tdf2(m))
          IF iArm = 1 THEN 'Include DF Raman # 1: A = T0 + Tdf1(m)
             Tdf1(m) = Trnd(tRam# + Tdf1(m))
             df$(m) = df$(m) + ";DT2,1," + STR$(Tdf1(m))
             fList$(nList - 3) = fList$(nList - 3) + Tdet$(Tdf1(m))
             IF Tdf1(m) < Tsh THEN Tsh = Tdf1(m) 'Tsh before Tdf1
          ELSE
             IF Tbl(m) < Tsh THEN Tsh = Tbl(m) 'Tsh before Tbl
          END IF
       END IF
       Tpr(m) = Trnd(tRam# + Tpr(m))
       pr$(m) = ";DT5,1," + STR$(Tpr(m)) 'probe: C = T0 + Tpr(m)
       fList$(nList) = fList$(nList) + Tdet$(Tpr(m))
       'Update list of pulse positions
       fList$(nList + 2) = fList$(nList + 2) + zFmt$(zmin(m))
       fList$(nList + 3) = fList$(nList + 3) + zFmt$(zmax(m))
       IF j > 0 THEN 'List beamsplitter positions
          FOR k = 0 TO 3
             fList$(nList + 4 + k) = fList$(nList + 4 + k) + zFmt$(zi(m, k))
          NEXT k
       END IF
    ELSE
       IF j > 0 AND NOT iINT(0) THEN 'Leave a blank column
          IF iArm = 1 THEN fList$(nList - 3) = fList$(nList - 3) + STRING$(14, 32)
          FOR k = 0 TO 2
             fList$(nList - k) = fList$(nList - k) + STRING$(14, 32)
          NEXT k
          FOR k = 2 TO 7
             fList$(nList + k) = fList$(nList + k) + STRING$(8, 32)
          NEXT k
       END IF
    END IF
    m = j + 1 'For j>0, store seq #m in mem #j
  NEXT j
  'Generate command for probe shutter and list the time
  IF NOT iFix THEN Tsh = Trnd(Tsh - .0025#)  'Round Tsh
  fList$(nFSK + 1) = "    Tsh:" + Tdet$(Tsh)
  'shutter: A = T0 + Tsh
  sh$ = ";DT2,1," + STR$(Tsh)

  iUp = 0 'Update AWFG, FF, Cs detuning and pulse times between launches?
  FOR j = 2 TO nSeq
    IF mSt(j) <> mSt(1) THEN iUp = -1 'More than one pattern occurs
  NEXT j

  'Print list of state selection/detection pulses and positions
  PRINT fList$(nFSK)
```

```
  FOR j = nFSK + 1 TO nListZ
    PRINT fList$(j)
    IF CSRLIN > 24 AND j < nListZ AND j <> nList THEN 'Fill screen with list
       PRINT : PRINT "Press any key to continue.";
       IF (NOT doak%) THEN DO WHILE INKEY$ = "": LOOP
       CLS
       IF j < nList THEN
          PRINT fList$(nFSK) 'Print detection pulse header
       ELSE
          PRINT fList$(nList + 1) 'Print header for positions
       END IF
    END IF
  NEXT j
  nList = nListZ

  IF (NOT doak%) THEN
    PRINT : INPUT "Are these frequencies satisfactory? (Y/n)"; a$
    IF UCASE$(a$) = "N" THEN iAuto = 0: iQuit = -1
  END IF
  CLS 'Prepare for list of download status information

  'Start the experiment pulsing
  outexpr = ";TM" + RIGHT$(STR$(iTrgMod), 1) + ";": l& = LEN(outexpr)
  CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
  IF nErr THEN PRINT "Trigger pulser: Error #"; nErr: iQuit = -1
  iTM = -1 'Experiment triggered

  'Set Raman start time on SRSP4: A = T0 + tRam#
  outexpr = ";DT2,1," + STR$(CSNG(tRam#)) + ";": l& = LEN(outexpr)
  CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
  IF nErr THEN PRINT "IEEE Error #"; nErr

  'Set the chrip rate:
  a$ = STR$(19000000# + CLNG(1000 * (1000000# + kg0# * .001 / 4)) / 1000)
  a$ = ";SPFR " + a$ + ";": l& = LEN(a$)
  CALL koutputStr(0, nErr, SSEGADD&(a$), l&, 0, mAdSnth4(0), mTerm(0))
  IF nErr THEN PRINT "Chirp UP dev: IEEE Error #"; nErr
  a$ = STR$(19000000# + CLNG(1000 * (1000000# - kg0# * .001 / 4)) / 1000)
  a$ = ";SPFR " + a$ + ";": l& = LEN(a$)
  CALL koutputStr(0, nErr, SSEGADD&(a$), l&, 0, mAdSnth5(0), mTerm(0))
'   IF nErr THEN PRINT "Chirp DOWN dev: IEEE Error #"; nErr

  IF iArb AND NOT iQuit THEN 'Download new gate pattern

    PRINT "Recalling default synthesizer settings."
    'Restore settings for gate signal
    outexpr = ";*SAV 9;*RCL 1;FSMP" + STR$(fSampG#): l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth1(0), mTerm(0))
    IF nErr THEN PRINT "Gate synth:Error #"; nErr: iQuit = -1
    'Set repetition rate for Cs lock detuning control (already initialized)
    outexpr = ":FREQ" + STR$(1 / (Tff# * nFF))
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
    IF nErr THEN PRINT "Cs synth: Error #"; nErr: iQuit = -1
    'Restore settings for Feed Forward control
    outexpr = HPInit$ + outexpr
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth2(0), mTerm(0))
    IF nErr THEN PRINT "FF synth: Error #"; nErr: iQuit = -1

    'Calculate checksum for gate contol
    isum& = 0
    FOR j = 0 TO iGi - 1
     isum& = isum& + iwfG(j)
    NEXT j
    isum& = isum& AND &HFFFF&
    iwfG(iGi) = isum& + (isum& > &H7FFF) * &H10000

    IF NOT iQuit THEN
       PRINT "Writing gate waveform."
```

```
        'Prepare SRS Synth #5 to receive arbitrary waveform
        outexpr = "LDWF?1," + LTRIM$(STR$(iGi / 2)): l& = LEN(outexpr)
        CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth1(0), mTerm(0))
        IF nErr THEN PRINT "Gate synth: Error #"; nErr: iQuit = -1

        'Wait for SRS Synth #5 to be ready for download
        inexpr = STRING$(30, 32)
        CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSnth1(0), mTerm(0))
        IF nErr THEN PRINT "Gate synth: Error #"; nErr: iQuit = -1
        IF VAL(inexpr) <> 1 THEN PRINT "Gate synthesizer not ready": iQuit = -1

        'Download gate waveform to SRS Synth #5
        CALL koutputBuf(0, nErr, iwfG(0), 2 * iGi + 2, 0, mAdSnth1(0), mTerm(0))
        IF nErr THEN PRINT "Gate synth: Error #"; nErr: iQuit = -1
     END IF
   END IF

   'Erase arrays no longer needed for frequency list
'    ERASE iwfG, v0#, z0, zmax, zmin, zi, Tdf1, Tbl, Tdf2, Tc#

   IF UCASE$(INKEY$) = "Q" THEN
     iQuit = -1
     doak% = 0
   END IF

   'Dynamically dimension arrays for AWFG pattern generation
   DIM if3(1000), if4(1000)

   IF NOT iQuit THEN 'Generate AWFG waveforms

     IF iArb THEN
        PRINT "Loading AO calibration tables."
        DEF SEG = VARSEG(if3(0)) 'F=3 AO
        BLOAD "E:\LAB\l3.bbn", VARPTR(if3(0))
        DEF SEG
        DEF SEG = VARSEG(if4(0)) 'F=4 AO
        BLOAD "E:\LAB\l4.bbn", VARPTR(if4(0))
        DEF SEG
        PRINT "Creating, downloading, and saving AWFG waveforms."
     END IF


     'Create any necessary amplitude patterns and download them
     y0 = 1 'Main pulse intensity at maximum
'*63b
     yA = 1
     yB = 1
'*63e
     nArb = 0 'Number of F=3 and 4 arbitrary patterns
     nArbFF = 0 'Number of FF patterns downloaded
     iyC = CSRLIN 'Base cursor line for reporting pattern status
     iyC2 = iyC - iArb 'Base cursor line for FF status
     m = 1 'Counter for pattern storage
     j = 0 'Counter for sequence #
     DO 'Step through sequences
        LOCATE iyC, 1
        IF iDwn(j) THEN 'Create AWFG pattern
           PRINT "Creating AWFG waveform for m ="
           LOCATE iyC, 32 + 2 * nArb: PRINT m
           'If AWFG pattern #j is already stored, then free memory
           nErr = XMSFree(j)
           'Allocate XMS memory for jth AWFG pattern
           nErr = XMSAlloc(j, lArb&)
           IF nErr THEN PRINT "XMSAlloc: Error #"; nErr
           'Select jth pattern for storing data
           nErr = XMSSelect(j)
           IF nErr THEN PRINT "XMSSelect: Error #"; nErr

           'The first sample is loaded when the START bit is set high,
           ' without waiting for the gate, so send extra point
           ia = 0: ib = 0: xDig% = 0 'All outputs OFF (STROBE active LOW)
```

```
'62:
           yDig% = CHPU1 OR CHPD1      'All outputs OFF (chirps active HIGH)
'           ia = 0: ib = 0: xDig% = 1 'All outputs OFF (STROBE active HIGH)
           iAWFG = -nAWFG0 'AWFG pattern pointer (0 -> 1st good sample)
           DO WHILE iAWFG < 0
              CALL AWFG(ia, ib)
           LOOP
           'Create preselection strobe
           ' Turn on beam switch and strobe
           xDig% = 4 + ik(m, 0) '0011->keff down, 0101->up; strobe active LOW
'           xDig% = 3 + ik(m, 0) '0010->keff down, 0100->up; strobe active HIGH
           FOR i = 1 TO iTs - 1 'Hold strobe active
              CALL AWFG(ia, ib)
           NEXT i
           ' Strobe will be turned off by first pulse

           'Set pi pulse times for preselection
           iTt1X = iTt1P: iTdX = iTdP: iTt2X = iTt2P: yX = yP
           FOR iFSK = 1 TO nFSK - 1
              IF iFSK = nPiP + 1 THEN 'Set pi pulse times for main pi's
                 iTt1X = iTt1: iTdX = iTd: iTt2X = iTt2: yX = y0
              END IF
              'Create digital output mask for AWFG Channel X
              xDig% = 3 + ik(m, iFSK - 1)
                 '2->keff down, 4->up; strobe OFF (active LOW)
'               xDig% = 4 + ik(m, iFSK - 1)
                 '0011->keff down, 0101->up; strobe OFF (active HIGH)
              IF ipl THEN
                 PRINT "Pulse"; iFSK; ": i0="; i0&(iFSK); "iAWFG="; iAWFG;
                 PRINT "ik="; iSt(m, iFSK); "xDig="; xDig%
              END IF
              IF iSt(m, iFSK - 1) = 0 THEN 'Atom in 3+4
                 iRev = (iSt(m, iFSK) = 1)
                 IF iRev THEN 'Pulse from 3+4 -> 4
'*63,*65b
                    yA = y0 * rtp!
                    yB = y0 * rbt!
'*63,*65e
                    CALL PulseSA(if4(), if3(), ik(m, iFSK - 1))
                 ELSE 'Pulse from 3+4 -> 3
'*63,*65b
                    yA = y0 * rbt!
                    yB = y0 * rtp!
'*63,*65e
                    CALL PulseSA(if3(), if4(), ik(m, iFSK - 1))
                 END IF
              ELSE
                 iRev = (iSt(m, iFSK - 1) = 1)
                 IF iRev THEN 'Atom in F=4
                    IF iSt(m, iFSK) = -1 THEN 'Pulse from 4 -> 3
                       CALL PulseAB(if4(), if3(), iTt1X, iTdX, iTt2X, yX)
                    ELSE 'Pulse from 4 -> 3+4
'*63b
                       yA = y0 * rtp!
                       yB = y0 * rbt!
'*63e
                       CALL PulseAS(if4(), if3(), ik(m, iFSK - 1))
                    END IF
                 ELSE 'Atom in F=3
                    IF iSt(m, iFSK) = 1 THEN 'Pulse from 3 -> 4
                       CALL PulseAB(if3(), if4(), iTt1X, iTdX, iTt2X, yX)
                    ELSE 'Pulse from 3 -> 3+4
'*63b
                       yA = y0 * rbt!
                       yB = y0 * rtp!
'*63e
                       CALL PulseAS(if3(), if4(), ik(m, iFSK - 1))
                    END IF
                 END IF
              END IF
           NEXT iFSK
```

```
           'Finish up end of AWFG pattern
           ia = 0: ib = 0
           DO WHILE iAWFG < lArb& - nAWFG0 - 1 'Count extra point at start
              CALL AWFG(ia, ib)
           LOOP
           xDig% = 0 'Turn off beam switch (STROBE active LOW)
'           xDig% = 1 'Turn off beam switch (STROBE active HIGH)
           CALL AWFG(ia, ib)
           IF iErr THEN PRINT "XMSPut2: Error #"; iErr
           iMem(j) = -1 'Set pattern storage flag for valid pattern
           nArb = nArb + 1
        END IF
        IF UCASE$(INKEY$) = "Q" THEN
          iQuit = -1
          doak% = 0
        END IF

        IF iINT(j) AND NOT iQuit THEN 'Download FF and Cs pattern
           IF CSRLIN > iyC2 THEN iyC2 = CSRLIN
           LOCATE iyC2, 1: PRINT "Writing feed forward pattern for m ="
           LOCATE iyC2, 37 + 2 * nArbFF: PRINT m
           outexpr = FFdata$ + FF$(m): l& = LEN(outexpr)
           CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth2(0), mTerm(0))
           IF nErr THEN PRINT "FF synth: Error #"; nErr: iQuit = -1

           iyC3 = CSRLIN
           PRINT "Writing Cs lock detuning pattern for m ="
           LOCATE iyC3, 41 + 2 * nArbFF: PRINT m
           outexpr = FFdata$ + Cs$(m)
           CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
           IF nErr THEN PRINT "Cs synth: Error #"; nErr: iQuit = -1

           'Save patterns as ARB_m
           outexpr = "DATA:COPY ARB_" + CHR$(48 + m): l& = LEN(outexpr)
           CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth2(0), mTerm(0))
           IF nErr THEN PRINT "FF synth: Error #"; nErr: iQuit = -1
           CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
           IF nErr THEN PRINT "Cs synth: Error #"; nErr: iQuit = -1

           nArbFF = nArbFF + 1 'Advance pattern counter
           IF UCASE$(INKEY$) = "Q" THEN
             iQuit = -1 'Don't store any more
             doak% = 0
           END IF
           IF ipl THEN
              PRINT "Press any key to continue."
              DO: a$ = UCASE$(INKEY$): LOOP WHILE a$ = ""
              iQuit = (a$ = "Q")
              CLS
           END IF
        END IF
        m = j + 1 'For j>0, store FF seq #m in mem #j
        j = j + 1 'Advance sequence
     LOOP WHILE j <= 4 AND NOT iQuit
   END IF

   'Erase arrays for FSK/AWFG pattern generation
   ERASE iDwn, iSt, ik, if3, if4, i0&, FF$, Cs$
'    ERASE if3, if4
'    ERASE FF$, Cs$

   IF iArm = -1 AND NOT iQuit THEN  'Eliminate the 1st DF Raman pulse
     outexpr = ";DT3,2,0": l& = LEN(outexpr) 'B = A + 0
     CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
     IF nErr THEN PRINT "DF Raman pulser:Error #"; nErr: iQuit = -1
   END IF

   IF iQuit GOTO 70

   'Collect data
```

```
    'Function Scan%() collects data and saves it in file Flnm$.
    '  Scan%() returns the value 0 if the routine finished collecting
    '   data, and -1 if the user Quit the routine prematurely.

    'Note the starting date and time for collecting data
    sDate$ = LEFT$(DATE$, 6) + RIGHT$(DATE$, 2): T1$ = TIME$
    iQuit = Scan%(mSt(), sh$, df$(), bl$(), pr$(), fCAve#(), fBack&(), nFSK0, iUp)
    IF iQuit THEN
'     iAuto = 0 'Display the menu
      doak% = 0
    ELSE
     IF iAuto <> 1 THEN 'Complete saving the file
      T2$ = TIME$ 'Stop time for data taking
      IF iAuto = 0 THEN 'Allow user to add comments
         BEEP: BEEP: BEEP
         LOCATE 7, 1: PRINT STRING$(19, 32): LOCATE 7, 1
         INPUT "Comments"; Cmmnt
      END IF

      'Parameters for Plot subroutine only
'      nSumm = 32 + 11 * iInt(0)
'*55      nSumm = 32 + 11 * iINT(0) + 1     'added line for the trigger mode
'*62      nSumm = 32 + 11 * iINT(0) + 2     'added line for chirp rate
      nSumm = 32 + 11 * iINT(0) + 3     'added line for imbalance factor
      IF (iauxmx > 0) THEN nSumm = nSumm + 1

      PRINT #1, nList; d; nSumm

      'Save times and frequencies of pulses
      FOR j = 0 TO nList
         PRINT #1, fList$(j)
      NEXT j

      'Save program parameters
      PRINT #1, "Data collected by AltInt71.BAS from "; T1$; " to "; T2$; " on "; sDate$
      IF (iauxmx > 0) THEN
        PRINT #1, "Auxillary device sequence"; iaux; " '"; AuxS$(iaux); "'"
      END IF
      PRINT #1, "Experiment trigger: "; TrgMod$(iTrgMod)
      PRINT #1, "Preselection offset ="; fOffP; "Hz with Raman trigger time = "; msec$(tRam#)
      PRINT #1, "Span ="; fspan; "Hz, Scan offset ="; fOff; "Hz"
      PRINT #1, LTRIM$(STR$(nRpt)); " repetitions at"; nSteps; "pts"
      PRINT #1, LTRIM$(STR$(nBlk)); " background samples detuned"; fBackOff; "Hz"
      PRINT #1, "Exit interferometers in state |"; CHR$(51 + (1 + iArmI) / 2); ">"
      PRINT #1, "DF Raman and blasting pulses are fixed "; Fix$(iFix)
      PRINT #1, "Cs lock detuning ="; fOffCs; "kHz, Gravitational acceleration g = "; g0#
      PRINT #1, "Chirp rate ="; kg0#; "Hz/s"
      PRINT #1, "Imbalance factors: TOP ="; rtp!; ", BOT ="; rbt!
      PRINT #1, "      Pi Pulse Parameters      "
      PRINT #1, "Delay time before starting the sequence = "; usec$(CDBL(Tstart))
      PRINT #1, LTRIM$(STR$(nPiP)); " "; usec$(iTpiP / fSamp#); " preselection pi pulses"
      PRINT #1, LTRIM$(STR$(nPi)); " "; usec$(iTpi / fSamp#); " main pi pulses spaced "; usec$(Tff#
* iFFpi); " apart"
      PRINT #1, "Fast transitions in "; usec$(iTt1P / fSamp#); "/"; usec$(iTt1 / fSamp#)
      PRINT #1, "Delays between turn-on and turn-off times = "; usec$(iTdP / fSamp#); "/"; usec$(iTd
 / fSamp#)
      PRINT #1, "Duration of frequency strobes = "; usec$(iTs / fSamp#)
      PRINT #1, "Delay between pulses and frequency strobes = "; usec$(iTdS / fSamp#)
      PRINT #1, "AWFG sample period = "; usec$(nCl / fCl)
      PRINT #1, "Rate parameters for pulse transitions ="; STR$(Cwf(0)); ","; Cwf(2)
      PRINT #1, "Equal intensity level is"; CINT(exp2 * xAmp); "% of full power"
      PRINT #1, "Relative intensity of preselection pulses is"; CINT(exp2 * yP); "% of full power"
      IF NOT iINT(0) THEN
         PRINT #1, "      Pi/2 Pulse Parameters      "
         PRINT #1, usec$(iTpiH / fSamp#); " pi/2 pulses spaced "; msec$(iTint& / fSamp#); " apart"
         PRINT #1, "Fast transitions in "; usec$(iTt1H / fSamp#); "/"; usec$(iTtyH / fSamp#)
         PRINT #1, "Delays between turn-on and turn-off times = "; usec$(iTdH / fSamp#)
         PRINT #1, "Time at equal intensities with common control on = "; usec$(iTon1 / fSamp#)
         PRINT #1, "Time at equal intensities with individual controls on = "; usec$(iTon2 / fSamp#)
         PRINT #1, "Rate parameters for pulse transitions ="; STR$(CwfH(0)); ","; CwfH(2)
         PRINT #1, "Equal intensity level is"; CINT(exp2 * xAmpH); "% of full phase"
```

```
        PRINT #1, "Phaselock settling time between pi/2's = "; usec$(iTpl / fSamp#)
        PRINT #1, "Minimum time for which tracer is OFF between pi/2's = "; usec$(CDBL(TbMin))
        PRINT #1, "Minimum time for which gate is OFF between pi/2's = "; usec$(CDBL(TgMin))
      END IF
      PRINT #1, "Comments:"; Cmmnt
      CLOSE #1
      PRINT "File saved."
      IF (FileCopy%(Flnm$, a$) = 0) THEN
        PRINT "File '"; Flnm$; "' copied to '"; a$; "'."
      ELSE
        PRINT "File '"; Flnm$; "' NOT copied!"
      END IF
     END IF
    END IF

70  'Erase the frequency list arrays
'    ERASE FSK&, fList$

    IF NOT iQuit THEN
      IF iAuto = 0 THEN 'Allow plotting and fitting of this data
        a$ = FlnmAdj$(Flnm$, "E:\LAB\DAT\", "DAT")
        CLS : CALL Plot(0, a$, psc)'Mode 0 -> Don't print frequency list
      END IF
      IF iAuto <> 1 THEN 'File has been saved -> Increment file number
        Flnm$ = IncFN$(Flnm$)
        CALL CheckName(Flnm$, 1)
        FilePrfx$ = LEFT$(Flnm$, 2)
      END IF
    END IF

    iQuit = 0 'Clear Quit flag
    GOTO 50 'Return to menu

    END

160   iErr = -1
      RESUME NEXT

9999  PRINT "Undetermined error."
      PRINT "Press any key to return to menu."
      DO: a$ = INKEY$: LOOP WHILE INKEY$ = ""
      RESUME 10000
10000 GOTO 70

REM $STATIC
SUB AWFG (ia, ib)
'        *****     Writes data to AWFG board     *****
'
'        This subroutine is called by the routines PulseAB, PulseAS,
'     and PulseSA to save x and y channel data for the AWFG patterns.  Those
'     routines send two integer values, but don't specify which is the x
'     value and which is the y value.  The common variable iRev indicates
'     whether the order of the two values are inverted.  The common variables
'     variables xDig% and yDig% control the digital outputs of the AWFG.
'        Once the order of the data is determined, it is written to XMS
'     memory using routine XMSPut2.
'

   CONST ASL4& = &H10

   'Determine the order of the data, shift the bits for the analog output
   ' to bits 15-4, and add the digital bits from 3-0
   IF iRev THEN
      x& = ASL4 * ib OR xDig%
      Y& = ASL4 * ia OR yDig%
   ELSE
      x& = ASL4 * ia OR xDig%
      Y& = ASL4 * ib OR yDig%
   END IF
   iErr = XMSPut2(x&, Y&) 'Use common error variable iErr to read remotely
   iAWFG = iAWFG + 1 'Increment point counter
```

```
     'Check for insufficient time for pi pulses
     TspMin = Tpi + TdS + Ts + TplPi + TgMax
     IF Tsp < TspMin THEN
      iErr = -1
      LOCATE 27, 1: PRINT "Requires at least "; usec$(CDBL(TspMin)); " spacing of pi pulses."; Clear
$;
      LOCATE 28, 1: PRINT "Press any key to continue.";
      DO: LOOP WHILE INKEY$ = ""
      LOCATE 28, 1: PRINT STRING$(30, 32);
      EXIT SUB
     END IF

     'Determine size of time steps for FF pattern.
     ' Try to set Tff to a submultiple of Tsp such that fewer than nFFmax
     ' points are needed.
     'Find upper limit on the number iFFpi of FF points per main pi pulse
     ' Include 1 extra at start and end to switch back to fCent
     iFFpi = INT((nFFmax - nPiP - 2) / (nPi + 2 * ii * Tint / Tsp + 1))
     IF iFFpi < 1 THEN iFFpi = 1 'Need at least one
     'It should not be necessary to have better resolution than Tffmin
     iFFpimax = INT(Tsp / Tffmin) + 1
     IF iFFpi > iFFpimax THEN iFFpi = iFFpimax
     DO 'Calculate exact number of FF steps required for this value of iFFpi
        'Choose Tff#*iFFpi as close to Tsp as possible
        Tff# = CLNG(fClFF * Tsp / iFFpi) / fClFF
        'Determine the number of FF steps for each pulse
        ' FF steps for initial strobe for preselection - (>=2 gate samples)
        iFFs = INT(Ts + 2 * TgMax / Tff#) + 1
        ' FF steps per preselection pi
        TspPmin = TpiP + TdS + Ts + TplPi + TgMax
        iFFpiP = INT(TspPmin / Tff#) + 1
        IF nPiP > 0 THEN 'Include extra time before 1st preselection pi
           '# of FF steps before preselection
           iFF0 = INT((Tstart + TspPmin) / Tff#) + 1 - iFFpiP
        ELSE
           IF iINT(0) THEN 'Include extra time before 1st main pi
              '# of FF steps before main pi
              iFF0 = INT((Tstart + TspMin) / Tff#) + 1 - iFFpi
           ELSE 'Include full time Tstart before pi/2's
              iFF0 = INT(Tstart / Tff# + .995)
           END IF
        END IF
        '# of FF steps for 1st and 3rd pi/2's
        iFFH1 = INT((Tpl + TpiH + Tint / 2 + TgMax) / Tff#) + 1
        '# of FF steps for 2nd and 4th pi/2's
        iFFH2 = INT((Tint / 2 + TpiH + TdS + Ts + TgMax) / Tff#) + 1
        'Total FF points - Add 1 at end for fCent
        nFF = iFFs + iFF0 + nPiP * iFFpiP + nPi * iFFpi + 2 * ii * (iFFH1 + iFFH2) + 1
        'Try to have less than nFFmax points
        iRpt = (nFF > nFFmax AND iFFpi > 1)
        iFFpi = iFFpi + iRpt 'Decrement iFFpi to decrease nFF
        'The HP33120A synthesizers require at least 8 points
        IF nFF < 8 THEN iRpt = -1: iFFpi = iFFpi - iRpt
     LOOP WHILE iRpt

     IF nFF > nFFmax AND iMode THEN 'Print warning message
        LOCATE 27, 1: PRINT "Warning! Requires"; nFF; "FF points -- only"; nFFmax; " recommended.";
 Clear$;
        FOR idel = 1 TO 100: NEXT idel
'        DO: LOOP WHILE INKEY$ = ""
     END IF

     IF pl THEN
        PRINT "iFFs="; iFFs; "iFF0="; iFF0; "iFFpiP="; iFFpiP; "iFFpi="; iFFpi;
        PRINT "iFFH1="; iFFH1; "iFFH2="; iFFH2; "nFF="; nFF; "Tff="; Tff#
     END IF

     'Determine sample rates for PCIP-AWFG/2 and SRS (gate signal).
     ' The goal is to achieve the highest possible resolution for the
     ' AWFG board, subject to the constraint that its sample rate must be an
```

```
END SUB

SUB BinCon (f#, fRndB&, fRnd#)
'The DDS-1EB synthesizer has accesible output frequencies of 25 MHz/2^32.
'  This subroutine calculates the binary representation of the exact output
'  frequency nearest to the desired frequency "freq".  The subroutine also
'  returns the actual value of the output frequency.
'

'CONST fstep = iMult * 5.820766091346741D-03    'Step size for multiplied DDS-1EB
CONST fstep = iMult * 1000000000# / 2147483648# 'Step size for ADS-431 clocking at 1 GHz

   fRndB& = CLNG((fCent + sgndet * f#) / fstep) 'Binary repr. of output freq
   fRnd# = sgndet * (fstep * fRndB& - fCent) 'Actual output frequency

END SUB

SUB Clock (iMode) 'iMode = -1 -> Print warning message for FF too long
' Clock -- Subroutine to determine the timing and resolution of the various
'      waveform patterns.
'         Since the feed forward (FF) control needs four different patterns--
'      one for each interferometer--it requires an HP synthesizer so that the
'      patterns can be stored in memory.  The HP synthesizers can only be
'      programmed in point mode, so to achieve reasonable download times, we
'      wish minimize the number of FF points.  The next lowest resolution
'      pattern is the gating waveform from the SRS synthesizer.  This can be
'      programmed in vector mode, so it can be divided into 16300 time steps,
'      but these time steps must cover the entire 300ms+ of the interferometer
'      patterns.  The AWFG patterns for the F=3, 4, 3+4 amplitude controls,
'      the beam switching control, and the frequency strobe have the highest
'      resolution, with over 32000 points spread out only over the time that
'      the gate signal is high.
'         For each pulse, this routine determines the maximum number of FF
'      points that will be required, then locates the pulses in the nearest
'      preceding block of gate pulses of sufficient duration.  It maximizes
'      the resolution of the AWFG pattern, subject to the constraint on the
'      maximum pattern length.
'
     CONST nFFmax = 500 'Total number of FF steps (limited for download time)
     CONST fClFF = 40000000# 'Clock rate (Hz) for HP synth (FF pattern)
     CONST fClG = 40000000# 'Clock rate for SRS synth (AWFG gate signal)
     CONST nSRS = 16300 'Maximum number of pts for SRS synthesizer
     CONST nAWFG = 32767 'Maximum number of pts for PCIP-AWFG
     'CONST nAWFG = 8191 'Maximum number of pts for PCIP-AWFG
     CONST Tffmin = .0002 'Minimum time step for FF pattern
     CONST TgMax = .00005 'Maximum gate sample size--safety buffer for rounding
     CONST TplPi = .0002 'Minimum Raman phaselock settling times before pi's
     CONST pl = 0              'Print calculated parameter values?

     iErr = 0 'Clear error flag
     IF pl THEN CLS 'Clear screen before listing parameters
     'Determine which interferometers occur in the sequence
     ERASE iINT 'Clear all waveform flags
     FOR j = 1 TO nSeq
      iINT(mSeq(j)) = -1 '-1 -> Interferometer occurs
     NEXT j
     ii = 1 + iINT(0) 'ii = 1 -> include pi/2's

     'Check for overflow of frequency list array
     nFreq = nPiP + nPi + 4 * ii + 1 '# of frequencies (+1 for fCent)
     IF nFreq > nFSKmax THEN
      iErr = -1
      LOCATE 27, 1: PRINT "Requires"; nFreq; "different frequencies -- only"; nFSKmax; " available."
; Clear$;
      LOCATE 28, 1: PRINT "Press any key to continue.";
      DO: LOOP WHILE INKEY$ = ""
      LOCATE 28, 1: PRINT STRING$(30, 32);
      EXIT SUB
     END IF
     IF pl THEN PRINT "nFreq="; nFreq
```

```
     ' integer multiple of the SRS sample rate.
     'Calculate lower limit for # of clock cycles per sample on AWFG
'*60     nCl = INT(fCl * (nPiP * TpiP + nPi * Tpi + 4 * ii * TpiH) / nAWFG) + 1
     nCl = INT(fCl * (nPiP * TpiP + nPi * Tpi + 4 * ii * .001) / nAWFG) + 1
     'If user requests lower resolution, then increase sample period
     IF nCl < nClMin THEN nCl = nClMin
     'To count samples for pi/2's, we must first determine whether the beam
     ' switching and gate are turned off between the pi/2 pulses.
     'Switch off the tracer if it can be safely turned back on after the
     ' midpoint between the pi/2's, and it can be off at least TbMin.
     iOffB = ((Tpl < Tint / 2) AND (Tint - Tpl - TdS > TbMin))
     dTon = ABS(xTon1 - xTon2) * TpiH 'Extra duration of pi/2's into center
     'Switch off the gate if it can be off at least TgMin.
     IF iOffB THEN
      'Gate OFF between 1st pi/2 and center?
      iOffG(1) = ((Tint / 2 - dTon > TgMin) AND (Tint / 2 - TdS - Ts > TgMin))
      'Gate OFF between strobe and beeam switch?
      iOffG(2) = (Tint / 2 - dTon - Tpl - Ts > TgMin)
      'Gate OFF between beam switch and 2nd pi/2?
      iOffG(3) = (Tpl - dTon > TgMin)
     ELSE
      'Gate OFF between 1st pi/2 and center?
      iOffG(1) = (Tint / 2 - dTon > TgMin)
      'Gate OFF between strobe and 2nd pi/2?
      iOffG(2) = (Tint / 2 - dTon - Ts > TgMin)
      iOffG(3) = 0 'Never three OFF times
     END IF
'*62:
     iOffG(1) = iOffG(1) AND (Tint / 2 - .001 + TpiH > TgMin)
       'if chirp for 2nd Pi/2 pulse must start before center, don't gate OFF
     DO 'Determine the exact number of AWFG points for this value of nCl
      'We first need the gate sample rate so we know how many extra points
      ' are needed around the AWFG pulses because of the worse resolution
      ' of the gate signal.
      'Calculate ratio of minimum gate sample period (Tff#*nFF/nSRS) to AWFG
      ' sample period, and round up to the next integer.
      iCl& = INT(Tff# * nFF / nSRS * fCl / nCl) + 1 '-> Tgate = Tawfg*iCl&
      'Since fClG is a multiple of fCl, it is possible to achieve this
      ' value of Tgate with an integral number of gate clock cycles.
      fSampG# = fClG / CINT(fClG * iCl& * nCl / fCl) 'Gate sampling rate
      fSamp# = fCl / nCl 'AWFG sampling rate
      'Count exact number of gate samples with gate ON.  This determines
      '  the exact number of samples for the AWFG patterns
      ' Calculate integer number of AWFG samples nearest to requested times
      '  At least one step is required for all transition times
      iTt1P = fSamp# * xTt1 * TpiP: IF iTt1P = 0 THEN iTt1P = 1
      iTt1 = fSamp# * xTt1 * Tpi: IF iTt1 = 0 THEN iTt1 = 1
      iTt1H = fSamp# * xTt1H * TpiH: IF iTt1H = 0 THEN iTt1H = 1
'*68:
      iTtyH = fSamp# * yTt1H * TpiH: IF iTtyH = 0 THEN iTtyH = 1
      '  At least two steps required for strobe pulse
      iTs = fSamp# * Ts + 1: IF iTs = 1 THEN iTs = 2
      'Non-transition times may be zero
      iTdP = CINT(fSamp# * xTd * TpiP)
      iTd = CINT(fSamp# * xTd * Tpi)
      iTdH = CINT(fSamp# * xTdH * TpiH)
      iTon1 = CINT(fSamp# * xTon1 * TpiH)
      iTon2 = CINT(fSamp# * xTon2 * TpiH)
      iTon = iTon2 + iTt1H
      idTon = iTon2 + iTt1H - iTon1 - 1 'Difference of iTon's
      IF idTon > 0 THEN iTon = iTon1 + 1
      iTdS = CINT(fSamp# * TdS) '# pts between pulses and strobe
      iTpl = CINT(fSamp# * Tpl) '# pts needed for phaselock settling
'*61      iTpl0 = CINT(fSamp# * (Tpl - .001 + TpiH))
'*64:
      iTpl0 = CINT(fSamp# * (Tpl - (.001 - TdS) + TpiH))
        '# pts needed for phaselock settling
'*61      iTcp = CINT(fSamp# * (.001 - TpiH))
'*64:
      iTcp = CINT(fSamp# * ((.001 - TdS) - TpiH))
        '# additional pts for chirp control signal
```

```
      iTint2& = CLNG(fSamp# * Tint / 2) '# pts between pi/2 and midpoint
      iTint& = 2 * iTint2& '# pts between pi/2's (must be even)
      'Spend remainder of time on slow transitions, but at least 1 point
      iTt2P = CINT(fSamp# * TpiP) - 2 * (iTt1P + iTdP): IF iTt2P < 1 THEN iTt2P = 1
      iTt2 = CINT(fSamp# * Tpi) - 2 * (iTt1 + iTd): IF iTt2 < 1 THEN iTt2 = 1
'*67      iTt2H = CINT(fSamp# * TpiH) - iTt1H - iTdH - iTon: IF iTt2H < 1 THEN iTt2H = 1
      iTt2H = CINT(fSamp# * TpiH) - iTtyH - iTdH - iTon: IF iTt2H < 1 THEN iTt2H = 1
      'Calculate total steps per pulse.
      ' These are the pulse durations used to calculate the center times
      '  and frequencies, so they shouldn't include TdS and Ts, when the
      '  light is off.
      iTpiP = 2 * (iTt1P + iTdP) + iTt2P: iTpi = 2 * (iTt1 + iTd) + iTt2
'*67      iTpiH = iTt1H + iTdH + iTt2H + iTon
      iTpiH = iTtyH + iTdH + iTt2H + iTon

      'Determine number of "ON" gating samples
      ' For pi pulses
      iGs = INT((iTs - .5) / iCl&) + 1 'First gate pulse for presel. strobe
      iGpiP = INT((iTpiP + iTdS + iTs - .5) / iCl&) + 1 'Preselection
      iGpi = INT((iTpi + iTdS + iTs - .5) / iCl&) + 1 'Main pi's
      ' For pi/2 pairs, first find the absolute position relative to center
      '  of all transitions in #AWFG samples.  These coordinates are rounded
      '  to gate samples to determine when the gate must be ON.  Later, the
      '  coordinates will be corrected for the OFF gate samples, thereby
      '  giving the correct coordinates in the actual AWFG waveform.
      IF NOT iINT(0) THEN
'*60         iAH&(0) = -iTint2& - iTpiH 'First sample of 1st pi/2
         iAH&(0) = -iTint2& - iTpiH - iTcp 'First sample of 1st pi/2
         iTadj = ABS(idTon): IF (iTadj < iTdS + 1) AND iOffB THEN iTadj = iTdS + 1
         iAH&(1) = -iTint2& + iTadj 'First sample after 1st pi/2
         iAH&(2) = 0 'First sample of strobe
         iAH&(3) = iTs 'First sample after strobe
'*60          iAH&(5) = iTint2& - iTpl 'First sample after beam switch ON
         iAH&(5) = iTint2& - iTpl0 'First sample after beam switch ON
         iAH&(4) = iAH&(5) - 1 'Last sample before beam switch
'*60          iAH&(6) = iTint2& - ABS(idTon) 'First sample of 2nd pi/2
'*66          iAH&(6) = iTint2& - iTcp - ABS(idTon) 'First sample of 2nd pi/2
         iAH&(6) = iTint2& - iTcp 'First sample of 2nd pi/2
         iAH&(7) = iTint2& + iTpiH + iTdS + iTs'First sample after 2nd pi/2
         'If intensity controls are on, turn on tracer also
         IF iAH&(4) > iAH&(6) THEN iAH&(4) = iAH&(6): iAH&(5) = iAH&(4) + 1
         'Round each of these AWFG coordinates to gate coordinates
         FOR j = 0 TO 7 STEP 2
            iGH(j) = INT(iAH&(j) / iCl&) 'Round down
            iGH(j + 1) = INT((iAH&(j + 1) - .5) / iCl&) + 1 'Round up
         NEXT j
         IF NOT iOffB THEN iGH(4) = iGH(6): iGH(5) = iGH(6) 'No beam switch
         'Calculate the total number of ON gate samples for pi/2 pair
         iGHtot& = iGH(7) - iGH(0) 'Total gate samples for pi/2 sequence
         FOR j = 1 TO 3 'Subtract samples with gate OFF
            'If after rounding, the gate isn't turned off anymore, then
            ' correct logical variables so the gate signal can be generated.
            iOffG(j) = (iOffG(j) AND (iGH(2 * j - 1) < iGH(2 * j)))
            iGHtot& = iGHtot& + iOffG(j) * (iGH(2 * j) - iGH(2 * j - 1))
         NEXT j
      END IF

      'Find total ON gate samples, which also gives total AWFG samples
      ' Add one extra for beam switching off at end
      nOnG& = iGs + nPiP * iGpiP + nPi * iGpi + 2 * ii * iGHtot& + 1
      lArb& = nOnG& * iCl& + nAWFG0 'Total length of AWFG pattern
      IF lArb& > nAWFG THEN nCl = nCl + 1
     LOOP WHILE lArb& > nAWFG

     'Find # AWFG points in pi/2 before common switch OFF
'*68     iTpiHC = iTpiH - iTon + iTon1 + 1
     iTpiHC = iTpiH - iTon + iTon1

     IF NOT iINT(0) THEN 'Correct AWFG coordinates iAH& for OFF gate samples
      iAH0& = iAH&(0) 'Save offset from center of 1st point for SUB Pi2Pi2
      iAH&(0) = 0 '1st pi/2 -> Take i0& (iAH0&) as origin
      iAH&(1) = iAH&(1) - iAH0&
'*66      iAH1& = iAH&(6) + ABS(idTon) '2nd pi/2 -> Take i0& (iAH1&) as origin
      iAH1& = iAH&(6) + iTcp    '2nd pi/2 -> Take i0& (iAH1&) as origin
      FOR i = 2 TO 7 'Step through coordinates iAH&(2)-iAH&(7)
         FOR j = 1 + i \ 2 TO 3 'Step through OFF periods
            iAH&(i) = iAH&(i) - iOffG(j) * iCl& * (iGH(2 * j) - iGH(2 * j - 1))
         NEXT j
         iAH&(i) = iAH&(i) - iAH1&
      NEXT i
     END IF

     IF pl THEN
         PRINT "iOffB="; iOffB; "iOffG="; iOffG(1); iOffG(2); iOffG(3)
         PRINT "iCl="; iCl&; "fSampG="; fSampG#; "fSamp="; fSamp#
         PRINT "iTs="; iTs; "iTdS="; iTdS; "iTpl="; iTpl; "iTint="; iTint&
         PRINT "iTpiP="; iTpiP; ":iTt1P="; iTt1P; "iTdP="; iTdP; "iTt2P="; iTt2P
         PRINT "iTpi="; iTpi; ":iTt1="; iTt1; "iTd="; iTd; "iTt2="; iTt2
         IF NOT iINT(0) THEN
            PRINT "iTpiH="; iTpiH; ":iTt1H="; iTt1H; "iTdH="; iTdH; "iTt2H="; iTt2H; "iTon1="; iTon1
; "iTon2="; iTon2; "iTon="; iTon
            PRINT "iTadj="; iTadj
            PRINT "iAH0="; iAH0&; "iAH1="; iAH1&
            PRINT "iAH()="; : FOR i = 0 TO 7: PRINT iAH&(i); : NEXT i: PRINT
            PRINT "iGH()="; : FOR i = 0 TO 7: PRINT iGH(i); : NEXT i: PRINT
            PRINT "iGHtot="; iGHtot&; "iTpiHC="; iTpiHC; "iTpiHB="; iTpiHB
         END IF
         PRINT "iGs="; iGs; "iGpiP="; iGpiP; "iGpi="; iGpi; "nOnG="; nOnG&; "lArb="; lArb&

         PRINT "Press any key to continue.": DO: LOOP WHILE INKEY$ = ""
         CLS
     END IF

END SUB

FUNCTION CsTBL% (f#)
'*****  Function to determine the HP synthesizer amplitude setting (-2047 to
'           +2047) required to provide a Cs lock AO tuning voltage that
'           achieves the requested offset frequency from 60 MHz - fOffCs, the
'           AO frequency required for the F=4 light to be on the DF resonance
'           including the correction for an offset fOffCs in the Cs lock.
'****WRONG****
'           The HP synthesizer should be set for 10 Vpp, with a 5 V offset.
'           The integer amplitude is expressed as two ASCII characters as
'           required for downloading arbitrary waveforms.
'****WRONG****
'           This code (orginally CsBin$()) assumed the HP Cs lock synthesizer
'           was set so that its amplitude = 11Vpp and its offset was set so
'           that the Cs lock was on resonance.
'*71b
'           While the lookup table still assumes 11Vpp, the code no longer
'           requires the synthesizer to be set that way.  In the global
'           variable "sCsAMP" it keeps track of the vertical scale the
'           synthesizer is set and scales the results of the table lookup
'           accordingly.
'*71e
'*70b
'   The absolute frequency of the two Raman beams is controlled by a single
' AO and is referenced to the saturation spectroscopy signal from a Cs cell.
' This Cs lock AO is double passed and thus shifts the pump beam from the
' probe beam by 2 x fAO, where fAO is the frequency of the RF to the AO with
' the sign of the shift included.  Since the pump and probe beam
' counterpropogate through the Cs cell, only atoms at rest will be resonant
' with both beams, and only if the absolute frequency of the laser is set to
' fCs - fAO, where fCs is the natural Doppler free transition frequency of
' the Cs line we are locking to.
'   Before we added the 120MHz switching AO, fAO was +60MHz, so the laser was
' locked to fCs - 60MHz.  Increasing the AO frequency fAO thus decreased the
' absolute laser frequency, hence the negative sign in fScCs.
'   To compensate for the 120MHz switching AO, we changed the sign of the Cs
' lock AO from +60MHz to -60MHz.  The laser will now be locked to fCs + 60MHz,
' and increasing the AO frequency will increase the absolute laser frequency.
'*70e
'*71b
'   The original function CsBin$() is now broken up into two functions: this
' one (CsTBL%()) and ItoS$(), which converts a 2-byte integer to a 2-character
' string of those two bytes.  Additionally, the result of the lookup table is
' now modified using two global variables "iCsOFF" and "sCsAMP".  "iCsOFF"
' is an integer in the same scale as the lookup table output which represents
' the current dc voltage offset of the HP synthesizer.  Before any non-zero
' lock offset (fOFFCs) was programmed into the arbitrary waveform pattern.
' Now, the dc offset value is changed, and consequently the output of the
' lookup table which represents the absolute frequency must have this offset
' value removed.  "iCsOFF" is that value that is subtract away.  "sCsAMP" is
' the current amplitude scale the HP synthesizer is set to (11.000 Vpp, for
' instance).  Because this amplitude is no longer constant, the program must
' keep track of what it is set to and rescale the output of the lookup table
' which assumed 11Vpp.
'*71e
'
'*69      CONST fScCs = -25000 'Scale frequency (in kHz) for table storage
      'Negative because fAO > 60 MHz shifts F=4 light below resonance
'*70:
      CONST fScCs = 25000 'Scale frequency (in kHz) for table storage
      'Positive becase increasing fAO increases the absolute laser frequency (see above)
      CONST pl% = 0

      'Convert f# to a scaled integer as used for the calibration table
      iFreq = CINT(&H8000& * (f# + fOffCs) / fScCs)

      IF iFreq < iCs(1, 0) OR iFreq > iCs(1, nDiv) THEN
         PRINT "Cs lock out of range.  Press any key to continue."
         DO: LOOP WHILE INKEY$ = ""
         IF iFreq < iCs(1, 0) THEN
            j = 0
         ELSE
            j = nDiv
         END IF
      ELSE 'Perform a binary search of the table for the frequency interval
           '  containing the desired frequency iFreq
         j = INT(nDiv / 2)
         jStep = INT((j + 1) / 2)
         DO WHILE jStep > 0
            IF pl% THEN PRINT "j="; j; "jStep="; jStep; "iFreq="; iFreq; "iCs="; iCs(1, j)
            IF iFreq < iCs(1, j) THEN
               j = j - jStep
               IF j < 0 THEN j = 0
            ELSE
               j = j + jStep
               IF j > nDiv THEN j = nDiv
            END IF
            jStep = INT((jStep - (jStep > 1)) / 2)
         LOOP
         'Make sure j is lower limit of range
         IF (iFreq < iCs(1, j) AND j > 0) OR (j = nDiv) THEN j = j - 1
         IF pl% THEN
            PRINT "j="; j; "jStep="; jStep; "iFreq="; iFreq; "iCs="; iCs(1, j)
         END IF
      END IF

      'Interpolate the table values to find the D/A setting iVout that gives
      '  the closest available frequency
      iFreq1 = iCs(1, j): iFreq2 = iCs(1, j + 1)
      v1 = iCs(0, j): v2 = iCs(0, j + 1)
      v% = (v1 * (iFreq2 - iFreq) + v2 * (iFreq - iFreq1)) / (iFreq2 - iFreq1)
'*71:
      v% = CINT((v% - iCsOFF) * 11! / sCsAMP)
      IF v% < -2047 OR v% > 2047 THEN
         PRINT "Extrapolation out of range.  Press any key to continue."
         DO: LOOP WHILE INKEY$ = ""
         IF v% < -2047 THEN
            v% = -2047
         ELSE
            v% = 2047
         END IF
      END IF

      CsTBL% = v%

      IF pl% THEN
         PRINT "iFreq1="; iFreq1; "iFreq2="; iFreq2; "v1="; v1; "v2="; v2; "v%="; v%
         PRINT "Press any key to continue."
         DO: a$ = UCASE$(INKEY$): LOOP WHILE a$ = ""
         IF a$ = "Q" THEN SYSTEM
      END IF

END FUNCTION

FUNCTION Fctr$ (r#)
'        *****     Rounds frequencies to the nearest 0.01 Hz     *****
'
'        Takes a real number in Hz and returns a string with the frequency
'     rounded to the nearest 0.01 Hz.  The string has the format used for
'     adiabatic transfer pulse center frequencies in fList$, the list of
'     pulse times and frequencies.
'
    a$ = LTRIM$(STR$(CLNG(ABS(exp2 * r#)))): l = LEN(a$)
    IF l < 3 THEN a$ = STRING$(3 - l, 48) + a$: l = 3
    IF r# < 0 THEN a$ = "-" + a$: l = l + 1
    Fctr$ = STRING$(13 - l, 32) + LEFT$(a$, l - 2) + "." + RIGHT$(a$, 2)

END FUNCTION

FUNCTION FFbin$ (f#)
'*****  Function to convert frequencies |f#|<=5 MHz to the arbitrary pattern
'           amplitude range -2047 to +2047.  The integer amplitude is then
'           expressed as two ASCII characters.
'
   CONST fSc = -yMax / 5000000# 'Scale frequency for FF patterns

   'Max. pattern amplitudes +-2047 <-> +-10 MHz shifts)
   FFb& = CLNG(fSc * f#) AND &HFFFF&
   'Convert FFb& to two bytes and express as ASCII characters
   Fl% = FFb& AND &HFF: fH% = INT(FFb& / &H100)
   FFbin$ = CHR$(fH%) + CHR$(Fl%) 'FF sequence data

END FUNCTION

FUNCTION ItoS$ (i%)
'Convert i% to two bytes and express as ASCII characters
'This function was originally part of the CsBin$() function.
  li& = CLNG(i%) AND &HFFFF&
  lsb% = li& AND &HFF
  msb% = INT(li& / &H100)
  ItoS$ = CHR$(msb%) + CHR$(lsb%) 'Cs detuning sequence data

      IF pl% THEN
         PRINT "li&="; li&; "lsb%="; lsb%; "msb%="; msb%
         PRINT "Press any key to continue."
         DO: a$ = UCASE$(INKEY$): LOOP WHILE a$ = ""
         IF a$ = "Q" THEN SYSTEM
      END IF
END FUNCTION

FUNCTION iwf (t, a, ifa(), b())
'        *****     Generates the shaping function for the pulses     *****
'
'     The pulse shape is generated using a 1001 element linearization table
'     ifa which has values corresponding to a 12-bit D/A conversion.
'
      IF t < .5 THEN
        u = exp3 * a * (EXP(b(0) * t) - 1) * b(1)
      ELSE
        u = exp3 * a * (1 - (EXP(b(2) * (1 - t)) - 1) * b(3))
```

```
      END IF
’      u = t * exp3      ’ use linear changes
      i = INT(u)
      IF i < 0 THEN i = 0
      IF i > 999 THEN i = 999

      ’Linearly interpolate between table elements
      iwf = (i + 1 - u) * ifa(i) + (u - i) * ifa(i + 1)

’      iwf = CINT(4095 * u / exp3) ’Allows direct look at shapes

END FUNCTION

FUNCTION msec$ (t#)
’        *****     Rounds times to the nearest 0.0001 ms     *****
’
’        Takes a real number in seconds and returns a string with the time
’     expressed in ms, rounded to the nearest 0.1 us.
’
   a$ = LTRIM$(STR$(CLNG(exp7 * t#))): l = LEN(a$)
   IF l < 5 THEN a$ = STRING$(5 - l, 48) + a$
   msec$ = LEFT$(a$, LEN(a$) - 4) + "." + RIGHT$(a$, 4) + " ms"

END FUNCTION

SUB Pi2Pi2 (iArmX)
’        *****      Generates a pair of pi/2 pulses        *****
’
’        This subroutine steps through the feed forward pattern, generating
’     that pattern along with the frequency list.  First, it determines the
’     nearest gate sample.  Then it calculates the pulse start times and
’     center times.  The center times determine the pulse frequencies and
’     feed forward settings.  The start times are saved in an array for
’     future use in generating the AWFG patterns.
’
CONST pl% = 0 ’Print diagnostic messages?

      iFSK = iFSK + 1 ’Increment pulse counter for 1st pi/2
      IF pl% THEN PRINT "Pulse"; STR$(iFSK); ":";
      ’Find gate sample starting closest to FF point between pi/2’s
      i1G = CINT(fSampG# * (iFF + iExt + iFFH1) * Tff#)
      i0G = i1G + iGH(0) ’1st ON gate sample for this pulse
      i0&(iFSK) = iCl& * i1G + iAH0&
        ’Start index for 1st sample of 1st pi/2 pulse
’*60      Tc#(1) = (i0&(iFSK) + iTpiH / 2#) / fSamp# ’Exact center time for 1st pi/2
      Tc#(1) = (i0&(iFSK) + iTcp + iTpiH / 2#) / fSamp#  ’Exact center time for 1st pi/2
’*66      i0&(iFSk + 1) = i0&(iFSk) + iTpiH + iTint& ’Starting index for 2nd pi/2
      i0&(iFSK + 1) = i0&(iFSK) + iTcp + iTpiH + iTint&
        ’Starting index for 2nd pi/2
      Tc#(2) = (i0&(iFSK + 1) + iTpiH / 2#) / fSamp# ’2nd pi/2
’*66      Tc#(2) = (i0&(iFSK + 1) + iTcp + iTpiH / 2#) / fSamp#  ’2nd pi/2
      ’List Tc for both pi/2’s
      fList$(iFSK) = "pi/2 pair:" + Tctr$(Tc#(1))
      fList$(iFSK + 1) = STRING$(10, 32) + Tctr$(Tc#(2))
      FOR m = 1 TO 4 ’Step through all possible patterns
         IF iINT(m) THEN ’This pattern appears in sequence
            ’Determine trajectory change for the two pi/2’s
            ’Recoil sign set by state and keff direction
            dv# = -ik(m, iFSK - 1) * iSt(m, iFSK - 1) * vR / 2
            ’Determine exit state for 2nd pi/2
            iSt(m, iFSK) = 0 ’Atoms enter superposition state
            IF iArmX = 0 THEN ’Choose branch to match physical trajectory
               IF (m AND 2) = 2 THEN ’Int’s #2 & 3 follow upper trajectory
                  iSt(m, iFSK + 1) = ik(m, iFSK - 1) ’Upper trajectory
               ELSE
                  iSt(m, iFSK + 1) = -ik(m, iFSK - 1) ’Lower trajectory
               END IF
            ELSE ’Force user selected exit state for 2nd pi/2 pair
               iSt(m, iFSK + 1) = iArmX
            END IF
            ik(m, iFSK) = ik(m, iFSK - 1) ’Same keff sign for both pi/2’s
```

```
            n1 = iExt + iFFH1 ’FF points for 1st pi/2
            FOR j = 0 TO 1 ’Calculate changes for both pi/2’s
               ’Calculate the adjusted probe, blasting and DF Raman times
               ’New position at pulse center
               dTc# = Tc#(j + 1) - Tc#(j)
               z0(m) = z0(m) + v0#(m) * dTc# - g0# * dTc# ^ 2 / 2
               ’Save positions for beam splitters
               zi(m, j + 2 * ABS(iArmX)) = z0(m)
               ’New velocity at pulse center
               vSgn = v0#(m) > 0 ’Remember initial sign of v0#(m)
               v0#(m) = v0#(m) - g0# * dTc#
               IF j = 0 THEN v0#(m) = v0#(m) + dv# ’Add recoil for 1st pi/2
               ’Update extremes of trajectory
               IF vSgn AND v0#(m) < 0 THEN ’Trajectory peaked between pulses
                  zpk = z0(m) + v0#(m) ^ 2 / 2 / g0# ’Peak height
                  IF zpk > zmax(m) THEN zmax(m) = zpk
               ELSE ’Check for peak at center time of pulse
                  IF z0(m) > zmax(m) THEN zmax(m) = z0(m)
               END IF
               IF z0(m) < zmin(m) THEN zmin(m) = z0(m) ’Check for minimum z
               IF NOT iFix THEN ’Calculate change of Tdf and Tbl
                  IF iArm = 1 THEN ’Include 1st DF Raman pulse
                     ’Adjusted DF Raman #1 time
                     Tdf1(m) = Tcorr(Tdf1(m), Tc#(j + 1), v0#(m), dv#)
                  END IF
                  ’Adjusted blowaway time
                  Tdf1(m) = Tcorr(Tbl(m), Tc#(j + 1), v0#(m), dv#)
                  ’Adjusted DF Raman time
                  Tdf2(m) = Tcorr(Tdf2(m), Tc#(j + 1), v0#(m), dv#)
               END IF
               ’Adjusted probe time
               Tpr(m) = Tcorr(Tpr(m), Tc#(j + 1), v0#(m), dv#)

               ’Calculate and list frequency of (j + 1)th pi/2
               fC#(m, j) = ik(m, iFSK) * keff * v0#(m) ’Includes Doppler shift and recoil
’*62b  Correct frequencies for chirp:
’               CALL BinCon(fC#(m, j), fRndB&, fRndCs#)
                ’If we are chirping only the F=3 light, don’t include the
                ’frequency offset in the Cs lock offset, since the freq of
                ’the F=4 hasn’t changed.  fRndCs# does not include this
                ’offset while fRnd# does.
’               fC#(m, j) = fC#(m, j) + ik(m, iFSK) * kg0# * (.001 - TpiH / 2)
’*62e
’*64:
               fC#(m, j) = fC#(m, j) - ik(m, iFSK) * kg0# * ((.001 - iTdS / fSamp#) - iTpiH / fSamp#
 / 2)
               ’Find binary representation of nearest accessible frequency
               CALL BinCon(fC#(m, j), fRndB&, fRnd#)
               fList$(iFSK + j) = fList$(iFSK + j) + Fctr$(fRnd#)
               FSK&(iFSK + j - 1, m) = fRndB&
’               IF pl% THEN PRINT (fCent + sgndet * fRnd#) / iMult;
               IF pl% THEN PRINT (fCent + sgndet * fRnd#);
               ’Update FF list
               b$ = FFbin$(fRnd#) ’New FF value
               c$ = ItoS$(CsTBL%(fRnd# / f21)) ’New Cs detuning value
’*62:
’               c$ = Csbin$(fRndCs# / f21) ’New Cs detuning value
               FOR N = 1 TO n1
                  FF$(m) = FF$(m) + b$
                  Cs$(m) = Cs$(m) + c$
               NEXT N
               n1 = iFFH2 ’FF points for 2nd pi/2

               ’Flip dv# to exit with no net recoil if leaving in same state
               IF j = 0 AND iSt(m, iFSK + 1) = iSt(m, iFSK - 1) THEN dv# = -dv#
            NEXT j

            v0#(m) = v0#(m) + dv# ’Add recoil from 2nd pi/2
            IF iArmX = 0 THEN ’Switch keff sign (1st pi/2 pair)
               ik(m, iFSK + 1) = -ik(m, iFSK)
            ELSE ’No change of keff sign (2nd pi/2 pair)
```

```
               ik(m, iFSK + 1) = ik(m, iFSK)
            END IF
         ELSE ’Leave space in frequency lists
            fList$(iFSK) = fList$(iFSK) + STRING$(14, 32)
            fList$(iFSK + 1) = fList$(iFSK + 1) + STRING$(14, 32)
         END IF
      NEXT m

      IF pl% THEN PRINT
      ’Update number of OFF gate samples before pi/2 sequence
      nOffG = nOffG + i0G - iG
      ’Correct i0& for the OFF gate samples
      i0&(iFSK) = i0&(iFSK) - nOffG * iCl& ’Start of 1st pi/2
      ’Update gate waveform for 1st pi/2
      iwfG(iGi) = iG: iwfG(iGi + 1) = 0: iGi = iGi + 2
      IF i0G - iG > 1 THEN iwfG(iGi) = i0G - 1: iwfG(iGi + 1) = 0: iGi = iGi + 2
      iwfG(iGi) = i0G: iwfG(iGi + 1) = yMax: iGi = iGi + 2
      iG = i0G ’Update gate sample pointer to start of 1st pi/2
      ’Update gate waveform for any gate=OFF times
      FOR j = 1 TO 3 ’Step through three possible OFF periods for gate
         IF iOffG(j) THEN ’Turn gate OFF and then back ON
            IF i1G + iGH(2 * j - 1) - iG > 1 THEN ’Repeat ON
               iwfG(iGi) = i1G + iGH(2 * j - 1) - 1: iwfG(iGi + 1) = yMax
               iGi = iGi + 2
            END IF
            iwfG(iGi) = i1G + iGH(2 * j - 1): iwfG(iGi + 1) = 0: iGi = iGi + 2
            IF iGH(2 * j) - iGH(2 * j - 1) > 1 THEN ’Repeat OFF
               iwfG(iGi) = i1G + iGH(2 * j) - 1: iwfG(iGi + 1) = 0
               iGi = iGi + 2
            END IF
            iwfG(iGi) = i1G + iGH(2 * j): iwfG(iGi + 1) = yMax
            iG = iwfG(iGi): iGi = iGi + 2
            ’Update number of OFF gate samples
            nOffG = nOffG + iGH(2 * j) - iGH(2 * j - 1)
         END IF
      NEXT j
      ’Finish up gate waveform for 2nd pi/2
      IF i1G + iGH(7) - iG > 1 THEN ’Repeat ON
         iwfG(iGi) = i1G + iGH(7) - 1: iwfG(iGi + 1) = yMax
         iGi = iGi + 2
      END IF
      iG = iwfG(iGi - 2) + 1 ’Update gate sample counter
      ’Correct i0& for 2nd pi/2 for the OFF gate samples
      i0&(iFSK + 1) = i0&(iFSK + 1) - nOffG * iCl& ’Start of 2nd pi/2

      PRINT fList$(iFSK) ’Print frequencies for 1st pi/2
      iFSK = iFSK + 1 ’Increment pulse counter for 2nd pi/2
      PRINT fList$(iFSK) ’Print frequencies for 2nd pi/2
      iFF = iFF + iExt + iFFH1 + iFFH2 ’Advance FF sequence pointer
      Tc#(0) = Tc#(2) ’Update pulse center time corresponding to v0#()

END SUB

SUB PiPulse (nPiX, iFFX, iGX, iTX)
’
’        This subroutine steps through the feed forward pattern, generating
’     that pattern along with the frequency list.  First, it determines the
’     previous gate sample.  Then it calculates the pulse start time and
’     center time.  The center time determines the pulse frequency and
’     feed forward setting.  The start time is saved in an array for future
’     use in generating the AWFG patterns.
’
CONST pl% = 0           ’Print diagnostic messages?

      iFSK = iFSK + 1 ’Advance pulse counter
      IF pl% THEN PRINT "Pulse"; STR$(iFSK); ":";
      ’Find 1st gate sample after end of gate block for this pulse
      i1G = INT(fSampG# * (iFF + iExt + iFFX) * Tff#)
      i0G = i1G - iGX ’1st "ON" gate sample for this pulse
      i0&(iFSK) = iCl& * i1G - iTX - iTdS - iTs ’1st AWFG sample for this pulse
      Tc#(1) = (i0&(iFSK) + iTX / 2#) / fSamp# ’Exact center time for this pulse
```

```
      ’List Tc
      fList$(iFSK) = "   pi #" + RIGHT$(STR$(nPiX), 2) + ":" + Tctr$(Tc#(1))

      m = 1 ’Pointer for pattern storage
      FOR j = 0 TO 4 ’Step through all possible patterns
         IF iINT(j) THEN ’This pattern appears in sequence
            ’Calculate the adjusted probe, blasting and DF Raman times
            iSt(m, iFSK) = -iSt(m, iFSK - 1) ’Atoms switch states
            ’Recoil sign set by state and keff direction
            dv# = -ik(m, iFSK - 1) * iSt(m, iFSK - 1) * vR
            ’New position at pulse center
            dTc# = Tc#(1) - Tc#(0)
            z0(m) = z0(m) + v0#(m) * dTc# - g0# * dTc# ^ 2 / 2
            ’New velocity at pulse center
            vSgn = v0#(m) > 0 ’Remember initial sign of v0#(m)
            v0#(m) = v0#(m) - g0# * dTc# + dv#
            ’Update extremes of trajectory
            IF vSgn AND v0#(m) < 0 THEN ’Trajectory peaked between pulses
               zpk = z0(m) + v0#(m) ^ 2 / 2 / g0# ’Peak height
               IF zpk > zmax(m) THEN zmax(m) = zpk
            ELSE ’Check for peak at center time of pulse
               IF z0(m) > zmax(m) THEN zmax(m) = z0(m)
            END IF
            IF z0(m) < zmin(m) THEN zmin(m) = z0(m) ’Check for minimum z
            IF NOT iFix THEN ’Calculate change of Tdf and Tbl
               IF iArm = 1 THEN ’Include 1st DF Raman pulse
                  ’Adjusted DF Raman #1 time
                  Tdf1(m) = Tcorr(Tdf1(m), Tc#(1), v0#(m), dv#)
               END IF
               ’Adjusted blowaway time
               Tbl(m) = Tcorr(Tbl(m), Tc#(1), v0#(m), dv#)
               ’Adjusted DF Raman time
               Tdf2(m) = Tcorr(Tdf2(m), Tc#(1), v0#(m), dv#)
            END IF
            ’Adjusted probe time
            Tpr(m) = Tcorr(Tpr(m), Tc#(1), v0#(m), dv#)

            ’Calculate frequency for pulse, including Doppler shift and recoil
            fC#(m, 0) = ik(m, iFSK - 1) * keff * (v0#(m) - dv# / 2)
            ’Find binary representation of nearest accessible frequency
            CALL BinCon(fC#(m, 0), fRndB&, fRnd#)
            fList$(iFSK) = fList$(iFSK) + Fctr$(fRnd#)
            FSK&(iFSK - 1, m) = fRndB&
’            IF pl% THEN PRINT (fCent + sgndet * fRnd#) / iMult;
            IF pl% THEN PRINT (fCent + sgndet * fRnd#);
            ’Update FF list
            b$ = FFbin$(fRnd#) ’New FF value
            c$ = ItoS$(CsTBL%(fRnd# / f21)) ’New Cs detuning value
            FOR N = 1 TO iExt + iFFX
               FF$(m) = FF$(m) + b$
               Cs$(m) = Cs$(m) + c$
            NEXT N

            IF iRev THEN ’Main pi’s -> switch keff sign
               ik(m, iFSK) = -ik(m, iFSK - 1)
            ELSE ’Preselection pi’s -> don’t switch keff sign
               ik(m, iFSK) = ik(m, iFSK - 1)
            END IF
         ELSE ’Leave space in frequency list
            IF j > 0 AND NOT iINT(0) THEN fList$(iFSK) = fList$(iFSK) + STRING$(14, 32)
         END IF
         m = j + 1 ’For j>0, store seq #m in mem #j
      NEXT j

      PRINT fList$(iFSK) ’Print frequencies for this pulse
      IF pl% THEN PRINT
      ’Update gate waveform
      iwfG(iGi) = iG: iwfG(iGi + 1) = 0: iGi = iGi + 2 ’OFF
      IF i0G - iG > 1 THEN iwfG(iGi) = i0G - 1: iwfG(iGi + 1) = 0: iGi = iGi + 2
      iwfG(iGi) = i0G: iwfG(iGi + 1) = yMax: iGi = iGi + 2 ’ON
      IF iGX > 1 THEN iwfG(iGi) = i1G - 1: iwfG(iGi + 1) = yMax: iGi = iGi + 2
```

```
      'Update number of OFF gate samples
      nOffG = nOffG + i0G - iG
      'Correct i0& for the OFF gate samples
      i0&(iFSK) = i0&(iFSK) - nOffG * iCl&
      iG = i1G 'Advance gate sample counter
      iFF = iFF + iExt + iFFX 'Advance FF pointer
      Tc#(0) = Tc#(1) 'Update pulse center time corresponding to v0#()

END SUB

SUB PulseAB (ifa(), ifb(), iTt1X, iTdX, iTt2X, yX)
'    *****  Generates an adiabatic transfer pulse from state a to b     *****
'
'    The pulse is generated using timing data from the calling program,
'    along with the shaping defined by the function iwf().  The data is
'    written to XMS memory by calling AWFG().  Other XMS routines will later
'    transfer the waveform from XMS memory to the PCIP-AWFG board.
'
      ia = 0: ib = 0 'Start with both beams off
      DO WHILE iAWFG < i0&(iFSK) 'Beams off until start of pulse
         CALL AWFG(ia, ib)
      LOOP
      xDig% = xDig% OR COM1 'Turn on common intensity control
      FOR j = 1 TO iTt1X 'Turn on F=b light first
         x = j / iTt1X
         ib = iwf(x, yX, ifb(), Cwf())
         CALL AWFG(ia, ib)
      NEXT j
      FOR j = 1 TO iTdX 'Wait before turning on F=a light
         CALL AWFG(ia, ib)
      NEXT j
      FOR j = 1 TO iTt2X 'Turn F=b off, F=a on
         x = j / iTt2X
         ia = iwf(x, yX, ifa(), Cwf())
         ib = iwf(1 - x, yX, ifb(), Cwf())
         CALL AWFG(ia, ib)
      NEXT j
      FOR j = 1 TO iTdX 'Wait before turning off F=a light
         CALL AWFG(ia, ib)
      NEXT j
      FOR j = 1 TO iTt1X - 1 'Turn off F=a light last
         x = j / iTt1X
         ia = iwf(1 - x, yX, ifa(), Cwf())
         CALL AWFG(ia, ib)
      NEXT j
      ia = 0: xDig% = xDig% AND COM0 'Turn off F=a and common control
      FOR j = 0 TO iTdS 'Wait before strobe pulse
         CALL AWFG(ia, ib)
      NEXT j
      'Turn on strobe
      xDig% = xDig% OR STR1     'STROBE active LOW
'      xDig% = xDig% AND STR0    'STROBE active HIGH
      FOR j = 1 TO iTs - 1 'Leave strobe active
         CALL AWFG(ia, ib)
      NEXT j
      'Strobe and beam switch will be changed by next pulse
      IF iAWFG MOD iCl& <> iCl& - 1 THEN PRINT "PulseAB Error: iAWFG mod"; iCl&; "="; iAWFG MOD iCl&

END SUB

'*61 SUB PulseAS (ifa(), ifb())
'62:
SUB PulseAS (ifa(), ifb(), ike)
'    *****  Generates an adiabatic pulse from |a> to |a>+|b>  *****
'*62          - direction given by "ike"
'
'    The pulse is generated using timing data from the calling program,
'    along with the shaping defined by the function iwf().  The data is
'    written to XMS memory by calling AWFG().  Other XMS routines will later
'    transfer the waveform from XMS memory to the PCIP-AWFG board.
'
```

```
CONST pl = 0 'Print diagnostic messages?

'*60       iC = i0&(iFSK) + iTpiHC 'AWFG index for common switch
      iC = i0&(iFSK) + iTcp + iTpiHC 'AWFG index for common switch
      iBS = iOffB AND (i0&(iFSK) + iAH&(1) - 1) 'AWFG index for beam switch
      ia = 0: ib = 0 'Start with both beams off
      DO WHILE iAWFG < i0&(iFSK)
         CALL AWFG(ia, ib)
      LOOP
'      PRINT iAWFG, "Chirp TRIG active"
'*62b
      'turn ON chirp control:
      IF (ike = 1) THEN
        yDig% = yDig% AND CHPU0       'keff UP
      ELSEIF (ike = -1) THEN
        yDig% = yDig% AND CHPD0  'keff DOWN
'*63:
        SWAP yA, yB
      END IF
      DO WHILE iAWFG < (i0&(iFSK) + iTcp)  'Beams off until start of pulse
         CALL AWFG(ia, ib)
      LOOP
'*62e
      xDig% = xDig% OR COM1 'Turn on common intensity control
'      PRINT iAWFG, "F=3&4 ON, start B ON"
      IF pl > 10 THEN
         PRINT "iBS="; iBS; "iAWFG="; iAWFG; "iOffB="; iOffB; "iC="; iC; "iInd="; iInd; ""
         PRINT "i0&="; i0&(iFSK)
         FOR j = 0 TO 7
           PRINT "i0&+iAH"; j; "="; i0&(iFSK) + iAH&(j)
         NEXT j
'          PRINT "xDig0="; xDig0%; "xDig="; xDig%
      END IF
'*67       FOR j = 1 TO iTt1H 'Turn on F=b light first
      FOR j = 1 TO iTtyH 'Turn on F=b light first
'*67          X = j / iTt1H
         x = j / iTtyH
'*62          ib = iwf(X, y0, ifb(), CwfH())
'*63:
         ib = iwf(x, yB, ifb(), CwfH())
         CALL AWFG(ia, ib)
      NEXT j
'      PRINT iAWFG, "B ON, hold"
      FOR j = 1 TO iTdH 'Wait before turning on F=a light
         CALL AWFG(ia, ib)
      NEXT j
'      PRINT iAWFG, "Start B down to 50%, A up to 50%"
      FOR j = 1 TO iTt2H 'Shift F=b down and F=a up to the 50% phase point
'*62          ia = iwf(.5 * j / iTt2H, y0, ifa(), CwfH())
'*63:
         ia = iwf(.5 * j / iTt2H, yA, ifa(), CwfH())
'*62          ib = iwf(1 - .5 * j / iTt2H, y0, ifb(), CwfH())
'*63:
         ib = iwf(1 - .5 * j / iTt2H, yB, ifb(), CwfH())
         CALL AWFG(ia, ib)
      NEXT j
      'Finish up pulse for individual AO's

'      PRINT iAWFG, "A,B @ 50%, hold"
      FOR j = 1 TO iTon2 'Keep both at present level
         IF iAWFG = iC THEN
           xDig% = (xDig% AND COM0) 'Turn off common control
'*62:
           yDig% = yDig% OR (CHPU1 OR CHPD1)    'Turn OFF chirp control
'            PRINT iAWFG, "F=3&4, Chirp TRIG OFF"
         END IF
         IF iAWFG = iBS THEN xDig% = (xDig% AND BS0) 'Turn off tracer
         CALL AWFG(ia, ib)
      NEXT j
'      PRINT iAWFG, "Start A,B OFF"
      FOR j = 1 TO iTt1H 'Turn both off together from the 50% phase point
```

```
         IF iAWFG = iC THEN
           xDig% = (xDig% AND COM0) 'Turn off common control
'*62:
           yDig% = yDig% OR (CHPU1 OR CHPD1)    'Turn OFF chirp control
'            PRINT iAWFG, "F=3&4, Chirp TRIG OFF"
         END IF
         IF iAWFG = iBS THEN xDig% = (xDig% AND BS0) 'Turn off tracer
         x = .5 * (1 - j / iTt1H)
'*62         ia = iwf(X, y0, ifa(), CwfH()): ib = iwf(X, y0, ifb(), CwfH())
'*63:
         ia = iwf(x, yA, ifa(), CwfH()): ib = iwf(x, yB, ifb(), CwfH())
         CALL AWFG(ia, ib)
      NEXT j
'      PRINT iAWFG - 1, "A,B OFF"

      DO WHILE iAWFG <= iC OR iAWFG <= iBS 'Wait to turn off tracer and common
         IF iAWFG = iC THEN
           xDig% = (xDig% AND COM0) 'Turn off common control
'*62:
'            PRINT iAWFG, "F=3&4, Chirp TRIG OFF"
           yDig% = yDig% OR (CHPU1 OR CHPD1)    'Turn OFF chirp control
         END IF
         IF iAWFG = iBS THEN xDig% = (xDig% AND BS0) 'Turn off tracer
         CALL AWFG(ia, ib)
      LOOP

      IF iAWFG <> i0&(iFSK) + iAH&(1) THEN
         PRINT "PulseAS Error: iAWFG="; iAWFG; ", End at"; i0&(iFSK) + iAH&(1)
         PRINT "Press any key to continue.": DO: LOOP WHILE INKEY$ = ""
      END IF
      IF pl THEN PRINT "->"; xDig%

END SUB

'*61 SUB PulseSA (ifa(), ifb())
'*62:
SUB PulseSA (ifa(), ifb(), ike)
'    *****  Generates an adiabatic pulse from |a> to |a>+|b>  *****
'*62          - direction given by "ike"
'
'    The pulse is generated using timing data from the calling program,
'    along with the shaping defined by the function iwf().  The data is
'    written to XMS memory by calling AWFG().  Other XMS routines will later
'    transfer the waveform from XMS memory to the PCIP-AWFG board.
'
CONST pl = 0 'Print diagnostic messages?

      'Beams off until start of pulse (Correct start time for indiv. controls)
'*68       iC = i0&(iFSK) + iTpiH - iTpiHC 'AWFG index for common switch
      iC = i0&(iFSK) + iTpiH - iTpiHC - 1 'AWFG index for common switch
'*66       iC = i0&(iFSK) + iTcp + iTpiH - iTpiHC 'AWFG index for common switch
      iInd = iC - idTon 'AWFG index for individual controls
      iBS = iOffB AND (i0&(iFSK) + iAH&(4)) 'AWFG index for beam switch

      xDig0% = xDig% 'Save setting of beam switch
      IF iOffB THEN xDig% = (xDig% AND BS0) 'Start with tracer OFF
      IF pl > 10 THEN
         PRINT "iBS="; iBS; "iAWFG="; iAWFG; "iOffB="; iOffB; "iC="; iC; "iInd="; iInd; ""
         PRINT "i0&="; i0&(iFSK)
         FOR j = 0 TO 7
           PRINT "i0&+iAH"; j; "="; i0&(iFSK) + iAH&(j)
         NEXT j
'          PRINT "xDig0="; xDig0%; "xDig="; xDig%
      END IF

      'Fill up to strobe pulse
      ia = 0: ib = 0
'*63:
      IF (ike = -1) THEN SWAP yA, yB
      DO WHILE iAWFG < i0&(iFSK) + iAH&(2)
'*62b
```

```
         IF (iAWFG = (i0&(iFSK) + iAH&(6))) THEN    'turn ON chirp control
           IF (ike = 1) THEN
             yDig% = yDig% AND CHPU0       'keff UP
           ELSEIF (ike = -1) THEN
             yDig% = yDig% AND CHPD0  'keff DOWN
           END IF
'            PRINT iAWFG, "Chirp TRIG active"
         END IF
'*62e
         CALL AWFG(ia, ib)
      LOOP
      'Create strobe pulse
      xDig% = xDig% OR STR1     'STROBE active LOW
'      xDig% = xDig% AND STR0    'STROBE active HIGH
      FOR i = 1 TO iTs - 1
'*62b
         IF (iAWFG = (i0&(iFSK) + iAH&(6))) THEN     'turn ON chirp control
           IF (ike = 1) THEN
             yDig% = yDig% AND CHPU0       'keff UP
           ELSEIF (ike = -1) THEN
             yDig% = yDig% AND CHPD0  'keff DOWN
           END IF
'            PRINT iAWFG, "Chirp TRIG active"
         END IF
'*62e
         CALL AWFG(ia, ib)
      NEXT i
      'Turn off strobe
      xDig% = xDig% AND STR0    'STROBE active LOW
'      xDig% = xDig% OR STR1     'STROBE active HIGH
      'Fill up to when individual controls turn on
      DO WHILE iAWFG < iInd
         IF iAWFG = iBS THEN xDig% = (xDig% OR xDig0%) 'Turn on tracer
         IF iAWFG = iC THEN
           xDig% = (xDig% OR COM1) 'Turn on common control
'            PRINT iAWFG, "F=3&4 OFF->ON"
         END IF
'*62b
         IF (iAWFG = (i0&(iFSK) + iAH&(6))) THEN     'turn ON chirp control
           IF (ike = 1) THEN
             yDig% = yDig% AND CHPU0       'keff UP
           ELSEIF (ike = -1) THEN
             yDig% = yDig% AND CHPD0  'keff DOWN
           END IF
'            PRINT iAWFG, "Chirp TRIG active"
         END IF
'*62e
         CALL AWFG(ia, ib)
      LOOP
'      PRINT iAWFG, "Start both ON to 50%"
      FOR j = 1 TO iTt1H 'Turn both up to the 50% phase point
         IF iAWFG = iBS THEN xDig% = (xDig% OR xDig0%) 'Turn on tracer
         IF iAWFG = iC THEN xDig% = (xDig% OR COM1) 'Turn on common control
         x = .5 * j / iTt1H
'*62         ia = iwf(X, y0, ifa(), CwfH()): ib = iwf(X, y0, ifb(), CwfH())
'*63:
         ia = iwf(x, yA, ifa(), CwfH()): ib = iwf(x, yB, ifb(), CwfH())
         CALL AWFG(ia, ib)
      NEXT j
'      PRINT iAWFG, "both @ 50%, hold"
      FOR j = 1 TO iTon2 'Both held constant
         IF iAWFG = iBS THEN xDig% = (xDig% OR xDig0%) 'Turn on tracer
         IF iAWFG = iC THEN xDig% = (xDig% OR COM1) 'Turn on common control
         CALL AWFG(ia, ib)
      NEXT j
'      IF pl > 10 THEN PRINT "->"; xDig%
      IF iAWFG <= iBS THEN
         PRINT "Error: Beam not switched on by iAWFG="; iAWFG
         PRINT "Press any key to continue.": DO: LOOP WHILE INKEY$ = ""
      END IF
      IF iAWFG <= iC THEN
```

```
        PRINT "Error: Common switch still off by iAWFG="; iAWFG
        PRINT "Press any key to continue.": DO: LOOP WHILE INKEY$ = ""
     END IF
'     PRINT iAWFG, "Start A OFF, B up to 100%"
     FOR j = 1 TO iTt2H 'Turn F=a off, F=b up to 1
'*62       ia = iwf(.5 * (1 - j / iTt2H), y0, ifa(), CwfH())
'*63:
        ia = iwf(.5 * (1 - j / iTt2H), yA, ifa(), CwfH())
'*62       ib = iwf(.5 * (1 + j / iTt2H), y0, ifb(), CwfH())
'*63:
        ib = iwf(.5 * (1 + j / iTt2H), yB, ifb(), CwfH())
        CALL AWFG(ia, ib)
     NEXT j
'     PRINT iAWFG, "A OFF, B @ 100%, hold"
     FOR j = 1 TO iTdH 'Wait before turning off F=b light
        CALL AWFG(ia, ib)
     NEXT j
'     PRINT iAWFG, "Start B OFF"
'*67      FOR j = 1 TO iTt1H - 1'Turn off F=b light last
     FOR j = 1 TO iTtyH - 1      'Turn off F=b light last
'*67       X = 1 - j / iTt1H
        x = 1 - j / iTtyH
'*62       ib = iwf(X, y0, ifb(), CwfH())
'*63:
        ib = iwf(x, yB, ifb(), CwfH())
        CALL AWFG(ia, ib)
     NEXT j
     ib = 0: xDig% = xDig% AND COM0 'Turn off F=b and common control
'*62:
     yDig% = yDig% OR (CHPU1 OR CHPD1)    'Turn OFF chirp control
'     PRINT iAWFG, "A, B, F=3&4, Chirp TRIG OFF"
     FOR j = 0 TO iTdS 'Wait before strobe pulse
        CALL AWFG(ia, ib)
     NEXT j
     'Turn on strobe
     xDig% = xDig% OR STR1     'STROBE active LOW
'     xDig% = xDig% AND STR0    'STROBE active HIGH
     FOR j = 1 TO iTs - 1      'Leave strobe active
        CALL AWFG(ia, ib)
     NEXT j
     'Strobe will be turned off, and beams switched by next pulse
     IF iAWFG <> i0&(iFSK) + iAH&(7) - 1 THEN
        PRINT "PulseSA Error: iAWFG="; iAWFG; ", End at"; i0&(iFSK) + iAH&(7) - 1
        PRINT "Press any key to continue.": DO: LOOP WHILE INKEY$ = ""
     END IF
END SUB

FUNCTION RdInit% (a$)
'        *****        Read in initialization data        *****
  fp0$ = FilePrfx$ 'Need to recheck file names?
  ON ERROR GOTO 160
  iErr = 0
  OPEN a$ FOR INPUT AS 1
  IF iErr THEN
     LOCATE 28, 1: PRINT a$; " not found. Press any key to continue.";
     DO: LOOP WHILE INKEY$ = ""
     LOCATE 28, 1: PRINT Clear$;
     RdInit% = -1
  ELSE
'*54      INPUT #1, FilePrfx$, fOffP, fspan, fOff, nSteps, nRpt, nBlk, fBackOff
     INPUT #1, b$, fOffP, fspan, fOff, nSteps, nRpt, nBlk, fBackOff
     INPUT #1, fOffCs, nSeq
     FOR j = 1 TO nSeq: INPUT #1, mSeq(j): NEXT j
     INPUT #1, iArmI, iFix, g0#, ysc, yysc, Tstart, nPiP, TpiP, nPi, Tpi, Tsp
     INPUT #1, xTt1, xTd, Ts, TdS, nClMin, Cwf(0), Cwf(2), xAmp, yP, Tint, TpiH
     INPUT #1, xTt1H, xTdH, xTon1, xTon2, CwfH(0), CwfH(2), xAmpH, Tpl
'*68: add yTt1H
     INPUT #1, TbMin, TgMin, tRam#, ak$, iTrgMod, kg0#, yTt1H
'*54      iMod = -1 'Set modification flag
'*65b
  rtp! = 1!       'Set intensity scaling factors to 1
  rbt! = 1!
'*65e

     'Calculate other pulse shape parameters
     Cwf(1) = xAmp / (EXP(Cwf(0) / 2) - 1)
     Cwf(3) = (1 - xAmp) / (EXP(Cwf(2) / 2) - 1)
     CwfH(1) = xAmpH / (EXP(CwfH(0) / 2) - 1)
     CwfH(3) = (1 - xAmpH) / (EXP(CwfH(2) / 2) - 1)

     RdInit% = 0
  END IF
  CLOSE 1
  ON ERROR GOTO 0

  IF FilePrfx$ <> fp0$ THEN 'Need to create and check file name
     'Add date to prefix to get filename
     sDate$ = LEFT$(DATE$, 6) + RIGHT$(DATE$, 2)
     Flnm$ = FilePrfx$ + LEFT$(sDate$, 2) + MID$(sDate$, 4, 2) + "00.DAT"
     IF fp0$ <> "" THEN CLS 'Clear when in Menu
     PRINT "Checking the filename."
     CALL CheckName(Flnm$, 0)
     FilePrfx$ = LEFT$(Flnm$, 2)
     IF fp0$ <> "" THEN CLS 'Clear when in Menu
  END IF

END FUNCTION

SUB SetCsOFF (f)
'   Set the Cs lock offset to "f".  First, use CsTBl%() to calculate what
' voltage is required to produce this frequency.  Try to set the dc offset
' of the synthesizer to this value.  Because if the design of the HP
' synthesizer, not all dc offsets are possible.  The dc offset must obey
' these two rules: 1) OFF + AMP/2 < 10V
'                  2) OFF <= 2*AMP
' where AMP is the amplitude setting (11.000 Vpp, for instance).  If required
' offset value violates these rules, first try to reduce the amplitude
' setting.  This will work up until OFF = 8.000 (and AMP = 4.000 Vpp), which
' is the absolute maximum.
'   Note that this means that the dc offset of the synthesizer "vCsOFF" does
' NOT always correspond to the offset frequency "fOffCs".  However, any
' leftover difference will automatically be taken care of in the arbitrary
' waveform pattern, so the actual lock frequencies during the pulses are
' still correct.

  vCsOFFold = iCsOFF / (2 * 2047) * 11 + vCsOFF0  ' previous offset voltage
  sCsAMPold = sCsAMP
'  PRINT "vCsOFF0 ="; vCsOFF0; ", fOFFCs ="; fOFFCs; ", f ="; f; "; iCsOFF = "; iCsOFF; "; sCsAMP =
"; sCsAMP
'  PRINT "vCsOFFold ="; vCsOFFold;
  fOffCs = f            ' new offset frequency
  '  In order to get the correct value from CsTBL%(), we must temporarily set
  'the offset and scaling to 0 and 11Vpp.
  sCsAMP = 11!          'lookup table assumes amplitude scaling of 11 Vpp
  iCsOFF = 0            'lookup table assumes no offset
  iCsOFF = CsTBL%(0#)
    'integer from Cs lookup table required to set Cs detuning to f
  vCsOFF = iCsOFF / (2 * 2047) * 11 + vCsOFF0
    'convert this integer to a real voltage
  IF ((vCsOFF + sCsAMP / 2) > 10!) THEN          'violates rule #1?
    sCsAMP = INT((10! - vCsOFF) * 2)    'Yes -> try lower amplitude setting
    IF (vCsOFF > sCsAMP * 2) THEN                'violates rule #2?
'  PRINT "iCsOFF = "; iCsOFF
      vCsOFF = 8!                       'Yes -> give up and set to max
      sCsAMP = 4!
      iCsOFF = (vCsOFF - vCsOFF0) * (2 * 2047) / 11
    END IF
    IF (sCsAMP <> sCsAMPold) THEN       'new value different from old?
                                        'Yes -> change amplitude to new value
      outexpr = ":FUNC:SHAP SIN;:VOLT" + STR$(sCsAMP) + ";:FUNC:SHAP USER"
        'output to SINE-wave mode
        'amplitde to "sCsAMP"
        'output back to ARB mode
'  PRINT "outexpr="; outexpr
     l& = LEN(outexpr)
     CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
     sCsAMPold = sCsAMP
    END IF
  END IF
  IF (vCsOFF <> vCsOFFold) THEN         'new value different from old?
                                        'Yes -> change dc offset to new value
    vstp = .1                                   'maximum voltage step size
    IF (vCsOFF < vCsOFFold) THEN vstp = vstp * -1         'step down
    N = INT((vCsOFF - vCsOFFold) / vstp)                   'number of steps
'  PRINT "vCsOFF0 ="; vCsOFF0; "fOffCs = "; fOFFCs; "; iCsOFF = "; iCsOFF; "; vCsOff = "; vCsOFF; ";
 sCsAMP = "; sCsAMP; ""
'  PRINT "outexpr="; outexpr
    FOR i = 1 TO N      'change offset in steps of vstp
      v = vCsOFFold + i * vstp    'so that lock does not see a large jump
      v = CINT(1000! * v) / 1000
      outexpr = ":VOLT:OFFS" + STR$(v)
        'offset to "v"
      l& = LEN(outexpr)
      CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
      FOR l& = 1 TO 20000: NEXT l&         'delay
    NEXT i
    'set offset to final value
    v = CINT(1000! * vCsOFF) / 1000
    outexpr = ":VOLT:OFFS" + STR$(v)
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
  END IF
  IF (sCsAMP <> sCsAMPold) THEN       'new value different from old?
                                        'Yes -> change amplitude to new value
    'First, program an 8 point pattern to VOLATILE memory to set the output
    'level to the offset value so that changing the amplitude scale does
    'not temporarily produce a pattern with large changes which might knock
    'out the lock
    'Set Trigger Source = Bus for triggering via software control
    outexpr = CsData$
    c$ = ItoS$(CsTBL%(0#)) 'New Cs lock setting
    FOR N = 1 TO 8 'Generate waveform description
      outexpr = outexpr + c$
    NEXT N
    'Select volatile arbitrary waveform for fast control of DC output level
    'Trigger pattern and tell HP to wait for completion of trigger
    outexpr = outexpr + CsSel$
    'Download waveform for this output level
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
    outexpr = ":FUNC:SHAP SIN;:VOLT" + STR$(sCsAMP) + ";:FUNC:SHAP USER"
      'output to SINE-wave mode
      'amplitde to "sCsAMP"
      'output back to ARB mode
'  PRINT "outexpr="; outexpr
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
  END IF

'         PRINT "Press any key to continue."
'         DO: a$ = UCASE$(INKEY$): LOOP WHILE a$ = ""
'         IF a$ = "Q" THEN SYSTEM
END SUB

FUNCTION Tcorr (T1, t0#, u0#, du#)
'     *****      Corrects trajectory for momentum recoils        *****
'
'        This function corrects times in the atomic trajectory for the effect
'     of photon recoils.  It considers an atom that experiences a recoil du#
'     at time t0#, when its velocity is u0#, and calculates the corrected time
'     Tcorr that an atom passes through a fixed point which it would cross
'     at time t1 if undeflected.
'
   dt = T1 - t0#: dt2 = dt - u0# / g0#
   Tcorr = T1 + SQR(dt2 ^ 2 + 2 * du# * dt / g0#) - dt2

END FUNCTION

FUNCTION Tctr$ (t#)
'         *****     Rounds times to the nearest 0.0001 ms     *****
'
'            Takes a real number in seconds and returns a string with the time
'        expressed in ms, rounded to the nearest 0.0001 ms.  The string has the
'        format used for adiabatic transfer pulse center times in fList$, the
'        list of pulse times and frequencies.
'
    a$ = LTRIM$(STR$(CLNG(exp7 * t#))): l = LEN(a$)
    IF l < 5 THEN a$ = STRING$(5 - l, 48) + a$: l = 5
    Tctr$ = STRING$(8 - l, 32) + LEFT$(a$, l - 4) + "." + RIGHT$(a$, 4) + " "

END FUNCTION

FUNCTION Tdet$ (t)
'         *****     Rounds times to the nearest 0.01 ms     *****
'
'            Takes a real number in seconds and returns a string with the time
'        expressed in ms, rounded to the nearest 0.01 ms.  The string has the
'        format used for detection pulse times in fList$, the list of pulse
'        times and frequencies.
'
    a$ = LTRIM$(STR$(CLNG(exp5 * t))): l = LEN(a$)
    IF l < 3 THEN a$ = STRING$(3 - l, 48) + a$: l = 3
    Tdet$ = STRING$(10 - l, 32) + LEFT$(a$, l - 2) + "." + RIGHT$(a$, 2) + STRING$(3, 32)

END FUNCTION

FUNCTION Trnd (t#)
'         *****     Rounds times to the nearest 0.01 ms     *****
'
'            Rounds a time in seconds to the nearest 0.01 ms.  This is used for
'        delay times sent to the SRS pulse generators.
'
   Trnd = CLNG(exp5 * t#) / exp5

END FUNCTION

FUNCTION usec$ (t#)
'         *****     Rounds times to the nearest 0.1 us     *****
'
'            Takes a real number in seconds and returns a string with the time
'        expressed in us, rounded to the nearest 0.1 us.
'
   a$ = LTRIM$(STR$(CLNG(exp7 * t#))): l = LEN(a$)
   IF l < 2 THEN a$ = "0" + a$: l = 2
   usec$ = LEFT$(a$, l - 1) + "." + RIGHT$(a$, 1) + " us"

END FUNCTION

FUNCTION zFmt$ (z)
'         *****     Rounds positions to the nearest 0.1 cm     *****
'
'            Takes a real number in meters and returns a string with the position
'        expressed in cm, rounded to the nearest 0.1 cm.  The string has the
'        format used for atomic positions in fList$.
'
    a$ = LTRIM$(STR$(CLNG(exp3 * z))): l = LEN(a$)
    zFmt$ = STRING$(7 - l, 32) + LEFT$(a$, l - 1) + "." + RIGHT$(a$, 1)

END FUNCTION
```

C.1.1 Menu.BAS

The *Menu* module of the AltInt code provides the operator interface. With this code, the operator can set all of the input parameters that determine exactly when and how the data are taken: the span and offset of the frequency scan, the interferometer sequence start time, the single-photon detuning, the number, shape, and duration of the adiabatic transfer pulses, the time between $\pi/2$-pulses, *etc.*

```
'       MenuXX.Bas
'  Menu subroutine for the AltIntXX.BAS program
'
'Revision History:
'10  4/13/98   Removed from AltInt56.BAS
'11  7/28/98   Add chirp rate, kg0#, to menu, option "h".  Also include kg0#
'             in paramaters stored to config file.
'               Move "iFix" option to "j", gravity option to "g", main plot
'             scale to "k", inset plot scale to "K".
'12 11/ 6/98   Add menu option 's' to allow the intensity imbalance factor
'             "rimb" to be changed from.
'13  2/ 2/99   Change menu option 's' to ask for both BOT and TOP intensity
'             scaling factors.
'15  3/ 3/99   Change menu option 'ph' to include the times for both
'             phase-sensitive and pure-state fast transitions.  Save the
'             new global variable "yTt1H" to config files.
'16  3/ 1/01   Change menu option 'c' to accomodate the Cs lock routine:
'             SetCsOFF(f!) which sets the Cs lock offset to f.  Remove
'             code which slowly changes output Cs lock synthesizer output
'             value and replace with a signle call to SetCsOFF().  Add a
'             call to SetCsOFF() after a configuration file is loaded
'             (option 'm') in case the Cs offset value changes.
'
'$INCLUDE: 'e:\lab\inc\nihpdas.inc'

DEFINT I-N
DEFSNG A-H, O-Z

DECLARE SUB CheckName (Flnm$, iMode)
DECLARE SUB menu (fi$)
DECLARE FUNCTION Clear$ ()
DECLARE FUNCTION CsTBL% (f#)
DECLARE FUNCTION FileCopy% (src$, dst$)
DECLARE FUNCTION FlnmAdj$ (Flnm$, path$, ext$)
DECLARE FUNCTION IncFN$ (filen$)
DECLARE FUNCTION ItoS$ (i%)
DECLARE FUNCTION msec$ (t#)
DECLARE FUNCTION RdInit% (a$)
DECLARE FUNCTION usec$ (t#)
'AutoKey functions:
DECLARE FUNCTION getak$ ()
DECLARE FUNCTION iptak$ (q$)

'*15:$INCLUDE: 'e:\lab\inc\altint.inc'
'*16:
'$INCLUDE: 'inc\altint.inc'

'$DYNAMIC

'Labels used by CheckName()
165   iErr = -1
      RESUME NEXT
'Labels used by FileCopy()
350 iErr = 1              'error opening input or output file
    RESUME NEXT
360 iErr = -1             'normal termination
    RESUME NEXT

REM $STATIC
SUB CheckName (Flnm$, iM)
'*********      Finds first available file name      *********
'
      ON ERROR GOTO 165
      iErr = 0
      iyC = CSRLIN
      DO
         OPEN "e:\lab\dat\" + Flnm$ FOR INPUT AS 1
         CLOSE 1
         IF iErr = 0 THEN
            IF (iM = 0) THEN LOCATE iyC, 1: PRINT Flnm$; " found"
            Flnm$ = IncFN$(Flnm$)
         END IF
      LOOP WHILE (iErr = 0) AND (UCASE$(INKEY$) <> "Q")

      IF (iM = 0) THEN PRINT Flnm$; " not found"
      ON ERROR GOTO 0

END SUB

FUNCTION Clear$
'     *****     Creates a string of spaces to the end of the screen
   Clear$ = STRING$(80 - POS(0), 32)

END FUNCTION

FUNCTION FileCopy% (src$, dst$)
'  Copy a srouce file "srcpath+src$" to destination file "dst$"=
'"dstpath+src$", where "srcpath" and "dstpath" are an internal
'constants.

CONST srcpath = "e:\lab\dat\"
CONST dstpath = "g:\hmdat\incoming\"

  iErr = 0
  ON ERROR GOTO 350
  f$ = srcpath + src$
  dst$ = dstpath + src$
'  PRINT f$; " --> "; dst$
  OPEN f$ FOR INPUT AS 50
  IF (iErr <> 0) THEN
    FileCopy% = 1
    CLOSE #50
    EXIT FUNCTION
  END IF
  OPEN dst$ FOR OUTPUT AS 51
  IF (iErr <> 0) THEN
    FileCopy% = 2
    CLOSE #50
    CLOSE #51
    EXIT FUNCTION
  END IF
  ON ERROR GOTO 360
  iErr = 0
  DO
    LINE INPUT #50, a$
    IF (iErr = 0) THEN PRINT #51, a$
  LOOP WHILE (iErr = 0)
  CLOSE #50
  CLOSE #51
  FileCopy% = 0
END FUNCTION

FUNCTION FlnmAdj$ (B$, path$, ext$)
'  *****     Adds path and extension to file names as needed     *****
'
   a$ = B$
   'Add extension if necessary
   j = 1: l = LEN(a$)
   DO WHILE MID$(a$, j, 1) <> "." AND j < l: j = j + 1: LOOP
   IF j = l THEN a$ = a$ + "." + ext$

   'Add path if necessary
   j = 1
   DO WHILE MID$(a$, j, 1) <> "\" AND j < l: j = j + 1: LOOP
   IF j = l THEN a$ = path$ + a$

   FlnmAdj$ = a$
END FUNCTION

FUNCTION getak$
'LOCATE 23, 1
'PRINT "doak%="; doak%; ", iak%="; iak%; ", ak$='"; ak$; "'"
```

```
IF doak% THEN
  a$ = MID$(ak$, iak%, 1)
  iak% = iak% + 1
  IF (a$ = "\") AND (iak% <= LEN(ak$)) THEN
    a$ = MID$(ak$, iak%, 1)
    iak% = iak% + 1
    SELECT CASE a$
     CASE "n": a$ = CHR$(13)  'newline
     CASE "t": a$ = CHR$(9)   'tab
    END SELECT
  END IF
  IF iak% > LEN(ak$) THEN doak% = 0
ELSE
  DO: a$ = INKEY$: LOOP WHILE a$ = ""
END IF
getak$ = LEFT$(a$, LEN(a$))

END FUNCTION

FUNCTION IncFN$ (f$)
  iFNbr = VAL(MID$(f$, 7, 2)) 'File number
  IF iFNbr < 99 THEN 'Increment file number
    IncFN$ = LEFT$(f$, 6) + RIGHT$(STR$(iFNbr + 101), 2) + RIGHT$(f$, 4)
  ELSE 'Increment file prefix
    a$ = LEFT$(f$, 2)
    IF (a$ = "ZZ") THEN
      a$ = "AA"
    ELSEIF (RIGHT$(a$, 1) = "Z") THEN
      a$ = CHR$(ASC(LEFT$(a$, 1)) + 1) + "A"
    ELSE
      a$ = LEFT$(a$, 1) + CHR$(ASC(RIGHT$(a$, 1)) + 1)
    END IF
    IncFN$ = a$ + MID$(f$, 3, 4) + "00" + RIGHT$(f$, 4)
  END IF
END FUNCTION

FUNCTION iptak$ (q$)
PRINT q$;
'LOCATE 23, 1
'PRINT "doak%="; doak%; ", iak%="; iak%; ", ak$='"; ak$; "'"
IF doak% THEN
  i = iak%
  l = LEN(ak$)
  DO
    WHILE ((i <= l) AND (MID$(ak$, i, 1) <> "\")): i = i + 1: WEND
    i = i + 2
  LOOP WHILE ((i <= l) AND (MID$(ak$, i - 1, 1) <> "n"))
  a$ = MID$(ak$, iak%, i - iak% - 2)
  IF (i > l) THEN
    doak% = 0
  ELSE
    iak% = i
  END IF
ELSE
  INPUT "", a$
END IF
iptak$ = LEFT$(a$, LEN(a$))

END FUNCTION

SUB menu (finit$)
'   *****      Menu subroutine for scan parameters     *****
'
'        The menu is divided into two parts.  First general parameters
'     for the interferometer selection and the frequency scan are
'     selected.  Then a submenu allows selection of specific values
'     for the adiabatic transfer pulse timing and parameters.
'
CONST fOffPmax = 5000000! 'Maximum detuning from resonance

DIM mSave(1 TO 4)
```

```
1000 CLS
1005 i1 = 5
     IF iModH THEN i1 = 1       'If pi/2's modified, all interferometers bad
     IF iMod THEN i1 = 0        'If pi's modified, all sequences bad
     FOR i = i1 TO 4: iMem(i) = 0: NEXT i 'Clear int. flags

     LOCATE 1, 1
     PRINT " 1   File name is "; Flnm$; Clear$;
     PRINT "2/@  decrement/INCREMENT filename by one"; Clear$;
     PRINT "3/#  run/CHANGE autokey string '"; ak$; "'"; Clear$;
     PRINT " 4   Raman sequence trigger time is "; msec$(tRam#); Clear$;
     PRINT " 5   Preselection offset frequency is"; CSNG(fOffP / exp3); "kHz"; Clear$;
     PRINT " 6   Frequency span is"; fspan; "Hz"; Clear$;
     PRINT " 7   Offset frequency for scan is"; fOff; "Hz"; Clear$;
     PRINT " 8   Number of points is"; nSteps; Clear$;
     PRINT " 9   Number of signal reads/step is"; nRpt; Clear$;
     PRINT " a   Number of background readings is"; nBlk; Clear$;
     PRINT " b   Raman beam detuning for background readings is"; CSNG(fBackOff / exp3); "kHz"; Clea
r$;
     PRINT " c   Cs lock detuning from DF resonance is"; fOffCs; "kHz"; Clear$;
     PRINT " d   Interferometer sequence (1-4):";
     FOR j = 1 TO nSeq: PRINT mSeq(j); : NEXT j: PRINT Clear$;
     PRINT " e   Waveform sequences in synthesizer memory (0-4):";
     FOR j = 0 TO 4
         IF iMem(j) THEN PRINT j;
     NEXT j
     PRINT Clear$;
     PRINT " f   Exit interferometers in state |"; CHR$(51 + (1 + iArmI) / 2); ">"; Clear$;
     PRINT " g   Gravitational acceleration, g ="; g0#; "m/s2"; Clear$;
     PRINT " h   Chirp rate during Pi/2 pulses is"; kg0#; "Hz/s"; Clear$;
     PRINT " i   Scan mode is "; Auto$(iAuto); Clear$;
     PRINT " j   DF Raman and blasting pulses are fixed "; Fix$(iFix); Clear$;
     PRINT "k/K  Scale multiplier for main/INSET screen plot is"; ysc; "/"; yysc; Clear$;
     PRINT "l/L  "; : IF iTM THEN PRINT "Stop";  ELSE PRINT "Start";
     PRINT " experiment triggering/SET to "; TrgMod$(iTrgMod); Clear$;
     PRINT Clear$;
     PRINT "m/M  Load/SAVE initial settings"; Clear$;
     PRINT " o   Plot a stored file"; Clear$;
     PRINT " p   Edit Raman pulse descriptions"; Clear$;
     PRINT "r/R  AuxDev next/DOWNLOAD:"; iaux; "/"; iauxmx; "'"; AuxS$(iaux); "'"; Clear$;
     PRINT " ESC   End session"; Clear$;
     LOCATE 27, 1: PRINT "Change any (# or RET)?"; Clear$;
'     DO: a$ = INKEY$: LOOP WHILE a$ = ""
     a$ = getak$
     LOCATE 27, 1

   SELECT CASE ASC(a$)
    CASE 27     'ESC
'*16:
     CALL SetCsOFF(0!)          'return Cs lock to 0 detuning
     nErr = XMSFree(-1)         'Free XMS memory allocated for AWFG patterns
     SYSTEM
    CASE 13     'CR
     EXIT SUB
    CASE 49     '1
'     INPUT "Enter file prefix or RET:"; a$
     a$ = iptak$("Enter file prefix or RET:")
     a$ = RTRIM$(LTRIM$(a$))
     IF LEN(a$) >= 2 THEN
         MID$(Flnm$, 1, 2) = UCASE$(LEFT$(a$, 2))
         MID$(Flnm$, 7, 2) = "00"
         CLS : CALL CheckName(Flnm$, 0): CLS 'Check file name
         FilePrfx$ = LEFT$(Flnm$, 2)
     END IF
    CASE 50     '2
     Flnm$ = LEFT$(Flnm$, 6) + RIGHT$(STR$(VAL(MID$(Flnm$, 7, 2)) + 99), 2) + RIGHT$(Flnm$, 4)
    CASE 64     '@
     Flnm$ = LEFT$(Flnm$, 6) + RIGHT$(STR$(VAL(MID$(Flnm$, 7, 2)) + 101), 2) + RIGHT$(Flnm$, 4)
    CASE 51     '3
     IF ak$ <> "" THEN
```

```
       doak% = -1     'enable autokey mode
       iak% = 1            'autokey pointer
     END IF
    CASE 35     '#
     a$ = iptak$("Enter new autokey sequence: ")
     IF a$ <> "" THEN ak$ = LEFT$(a$, LEN(a$))
    CASE 52     '4
'     INPUT "Enter Raman sequence trigger time (#ms or RET):"; a$
     a$ = iptak$("Enter Raman sequence trigger time (#ms or RET):")
     ar = VAL(a$) / exp3
     IF a$ <> "" AND ar > 0 THEN
         LOCATE 27, 1: PRINT "Adjust preselection offset frequency (Y/n)?"; Clear$;
'          DO: a$ = UCASE$(INKEY$): LOOP WHILE a$ = ""
         a$ = UCASE$(getak$)
         IF a$ <> "N" THEN 'Check whether this tRam# gives a reasonable fOffP
            xSave = fOffP: fOffP = fOffP - keff * g0# * (ar - tRam#)
            IF ABS(fOffP) < fOffPmax THEN
               tRam# = ar
            ELSE
               fOffP = xSave
            END IF
         ELSE 'Change tRam without adjusting fOffP
            tRam# = ar
         END IF
     END IF
    CASE 53     '5
'     INPUT "Enter preselection offset frequency (#kHz or RET):"; a$
     a$ = iptak$("Enter preselection offset frequency (#kHz or RET): ")
     ar = exp3 * VAL(a$)
     IF a$ <> "" AND ABS(ar) < fOffPmax THEN fOffP = ar
    CASE 54     '6
'     INPUT "Enter frequency span (#Hz or RET):"; a$
     a$ = iptak$("Enter frequency span (#Hz or RET): ")
     IF a$ <> "" THEN fspan = VAL(a$)
    CASE 55     '7
'     INPUT "Enter scan offset frequency (#Hz or RET):"; a$
     a$ = iptak$("Enter scan offset frequency (#Hz or RET): ")
     IF a$ <> "" THEN fOff = VAL(a$)
    CASE 56     '8
'     INPUT "Enter number of points in scan (# or RET):"; a$
     a$ = iptak$("Enter number of points in scan (# or RET): ")
     IF a$ <> "" THEN nSteps = VAL(a$)
    CASE 57     '9
'     INPUT "Enter number of signal reads/step (# or RET):"; a$
     a$ = iptak$("Enter number of signal reads/step (# or RET): ")
     IF a$ <> "" THEN nRpt = VAL(a$)
     IF nRpt < 1 THEN nRpt = 1
    CASE 97     'a
'     INPUT "Enter number of background readings (# or RET):"; a$
     a$ = iptak$("Enter number of background readings (# or RET): ")
     IF a$ <> "" THEN nBlk = VAL(a$)
     IF nBlk < 2 THEN nBlk = 2
    CASE 98     'b
'     INPUT "Enter Raman beam detuning for background measurements (#kHz or RET):"; a$
     a$ = iptak$("Enter Raman beam detuning for background measurements (#kHz or RET): ")
     IF a$ <> "" THEN fBackOff = exp3 * VAL(a$)
    CASE 99     'c
'     INPUT "Enter Cs lock detuning (#kHz or RET):"; a$
     a$ = iptak$("Enter Cs lock detuning (#kHz or RET): ")
     IF a$ <> "" THEN
'*15         fOff0 = fOffCs 'Old offset
         fOffCs = CINT(VAL(a$)) 'New offset
'*16:
         CALL SetCsOFF(fOffCs)
'*15b
'         'Gradually shift the Cs detuning to fOffCs
'         FOR i = nFstep - 1 TO 0 STEP -1
'            fout# = (fOff0 - fOffCs) * i / nFstep
'            'Create waveform to set tuning voltage
'            c$ = ItoS$(CsTBL(fout#)) 'New Cs lock setting
'            outexpr = CsData$
```

```
'            FOR n = 1 TO 8
'               outexpr = outexpr + c$
'            NEXT n
'            'For first step, set Trig. Source = Bus and select VOLATILE arb
'            IF i = nFstep - 1 THEN outexpr = ":TRIG:SOUR BUS;" + outexpr + CsSel$
'            'Send trigger and wait for completion
'            outexpr = outexpr + Trig$
'            'Download waveform for this output level
'            l& = LEN(outexpr)
'            CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
'            IF nErr THEN PRINT "Cs synth: Error #"; nErr
'         NEXT i
'*15e
     END IF
    CASE 100    'd
'     INPUT "List up to 4 interferometers in desired order (1,2,3,4; 0 -> none):"; a$
     a$ = iptak$("List up to 4 interferometers in desired order (1,2,3,4; 0 -> none): ")
     IF a$ <> "" THEN
         FOR i = 1 TO 4
            mSave(i) = mSeq(i): mSeq(i) = 0 'Save old sequence then clear
         NEXT i
         nSave = nSeq: nSeq = 0: i = 1
         DO WHILE i <= LEN(a$) AND nSeq < 4
            ia = VAL(MID$(a$, i, 1))
            IF ia > 0 AND ia < 5 THEN nSeq = nSeq + 1: mSeq(nSeq) = ia
            i = i + 1
         LOOP
         IF nSeq = 0 THEN nSeq = 1
         CALL Clock(-1) 'Check nFreq, nFF limits with change of sequences
         IF iErr THEN  'Restore previous sequence
            FOR i = 1 TO 4: mSeq(i) = mSave(i): NEXT i
            nSeq = nSave
            CALL Clock(0)
         END IF
     END IF
    CASE 101    'e
'     INPUT "List the valid waveforms in memory (0-4):"; a$
     a$ = iptak$("List the valid waveforms in memory (0-4): ")
     iMod = 0: iModH = 0      'Clear modification flags
     IF a$ <> "" THEN
         FOR i = 0 TO 4: iMem(i) = 0: NEXT i    'Clear all memory flags
         i = 1
         DO WHILE i <= LEN(a$)
            ia = VAL(MID$(a$, i, 1))
            IF ia > 0 AND ia < 5 THEN
              'Set memory flag if pattern #ia exists in XMS memory
              IF XMSSelect(ia) = 0 THEN iMem(ia) = -1
            END IF
            i = i + 1
         LOOP
     END IF
    CASE 102    'f
     iArmI = -iArmI: iModH = -1
    CASE 103    'g
'     INPUT "Enter gravitational acceleration (#m/s^2 or RET):"; a$
     a$ = iptak$("Enter gravitational acceleration (#m/s2 or RET): ")
     ar# = VAL(a$): IF a$ <> "" AND (ar# >= 0) THEN g0# = ar#
    CASE 104    'h
     a$ = iptak$("Enter chirp rate (#Hz/s or RET): ")
     ar# = VAL(a$): IF a$ <> "" THEN kg0# = ar#
    CASE 105    'i
     iAuto = (iAuto + 1) MOD 3
    CASE 106    'j
     iFix = NOT iFix
    CASE 107    'k
'     INPUT "Enter main scale multiplier (# or RET):"; a$
     a$ = iptak$("Enter main scale multiplier (# or RET): ")
     ar = VAL(a$): IF ar <> 0 THEN ysc = ar
    CASE 75     'K
'     INPUT "Enter inset scale multiplier (# or RET):"; a$
     a$ = iptak$("Enter inset scale multiplier (# or RET): ")
```

```
      ar = VAL(a$): IF ar <> 0 THEN yysc = ar
     CASE 108    'l
'      outexpr = ";TM;": l& = LEN(outexpr) 'Request present trigger mode
'      CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
'      IF nErr THEN PRINT "IEEE Error #"; nErr
'      inexpr = STRING$(30, 32) 'Read in trigger mode
'      CALL kenterStr(0, nErr, SSEGADD&(inexpr), 30, 0, mAdSRSP3(0), mTerm(0))
'      IF nErr THEN PRINT "IEEE Error #"; nErr
'      iTM = (VAL(inexpr) <> 2)
      'Toggle between trigger mode set by iTrgMod and SingleShot (TM2)
      iTM = NOT iTM 'Flip state of iTM
      IF iTM THEN
        outexpr = ";TM" + RIGHT$(STR$(iTrgMod), 1) + ";"
      ELSE
        outexpr = ";TM2;"
      END IF
      l& = LEN(outexpr)
      CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
     CASE 76     'L
      IF iTrgMod = 0 THEN
        iTrgMod = 1
      ELSEIF iTrgMod = 1 THEN
        iTrgMod = 4
      ELSEIF iTrgMod = 4 THEN
        iTrgMod = 0
      END IF
      IF iTM THEN
        outexpr = ";TM" + RIGHT$(STR$(iTrgMod), 1) + ";"
        l& = LEN(outexpr)
        CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP3(0), mTerm(0))
      END IF
     CASE 109    'm
      PRINT "Enter file to load (RET for "; finit$; " or Q to quit): ";
'      INPUT a$: a$ = UCASE$(a$)
      a$ = UCASE$(iptak$(""))
      IF a$ <> "Q" THEN
       IF a$ = "" THEN a$ = finit$
       a$ = FlnmAdj$(a$, pinit, "INI")
       IF (RdInit%(a$) = 0) THEN
         CALL Clock(0)    'Recalculate waveform parameters
         iMod = -1
'*16:
         CALL SetCsOFF(fOffCs)
       END IF
      END IF
      iak% = 1            'autokey pointer
     CASE 77    'M
      PRINT "Enter file to save (RET for "; finit$; " or Q to quit): ";
'      INPUT a$: a$ = UCASE$(a$)
      a$ = UCASE$(iptak$(""))
      IF a$ <> "Q" THEN
       IF a$ = "" THEN a$ = finit$
       a$ = FlnmAdj$(a$, pinit, "INI")
       'Write data
       OPEN a$ FOR OUTPUT AS 1
       PRINT #1, FilePrfx$; d; fOffP; d; fspan; d; fOff; d; nSteps; d; nRpt
       PRINT #1, nBlk; d; fBackOff; d; fOffCs; d; nSeq
       FOR j = 1 TO nSeq: PRINT #1, mSeq(j): NEXT j
       PRINT #1, iArmI; d; iFix; d; g0#; d; ysc; d; yysc; d; Tstart; d; nPiP
       PRINT #1, TpiP; d; nPi; d; Tpi; d; Tsp; d; xTt1; d; xTd; d; Ts; d; TdS
       PRINT #1, nClMin; d; Cwf(0); d; Cwf(2); d; xAmp; d; yP; d; Tint; d; TpiH
       PRINT #1, xTt1H; d; xTdH; d; xTon1; d; xTon2; d; CwfH(0); d; CwfH(2)
       PRINT #1, xAmpH; d; Tpl; d; TbMin; d; TgMin; d; tRam#; d; ak$
'*14       PRINT #1, iTrgMod; d; kg0#
       PRINT #1, iTrgMod; d; kg0#; d; yTt1H
       CLOSE 1
      END IF
     CASE 111    'o
      CLS : LOCATE 1, 1
'      INPUT "Name of file to be plotted is "; FlnmP$: FlnmP$ = UCASE$(FlnmP$)
'      a$ = UCASE$(iptak$("Name of file to be plotted is "))
      IF (FlnmP$ = "") THEN FlnmP$ = LEFT$(Flnm$, 6) + RIGHT$(STR$(VAL(MID$(Flnm$, 7, 2)) + 99), 2) +
 RIGHT$(Flnm$, 4)
      PRINT "Name of file to be plotted is (";
      PRINT USING "\          \"; FlnmP$; : PRINT "):";
      DO
        DO: a$ = INKEY$: LOOP WHILE (a$ = "")
        i = ASC(a$)
        IF (i = 0) THEN
          SELECT CASE (ASC(MID$(a$, 2, 1)))
           CASE 75       'left arrow
            FlnmP$ = LEFT$(FlnmP$, 6) + RIGHT$(STR$(VAL(MID$(FlnmP$, 7, 2)) + 99), 2) + RIGHT$(FlnmP$
, 4)
           CASE 77       'right arrow
            FlnmP$ = LEFT$(FlnmP$, 6) + RIGHT$(STR$(VAL(MID$(FlnmP$, 7, 2)) + 101), 2) + RIGHT$(FlnmP
$, 4)
           CASE 115      'CTRL-left arrow
            FlnmP$ = LEFT$(FlnmP$, 6) + RIGHT$(STR$(VAL(MID$(FlnmP$, 7, 2)) + 90), 2) + RIGHT$(FlnmP$
, 4)
           CASE 116      'CTRL-right arrow
            FlnmP$ = LEFT$(FlnmP$, 6) + RIGHT$(STR$(VAL(MID$(FlnmP$, 7, 2)) + 110), 2) + RIGHT$(FlnmP
$, 4)
          END SELECT
          LOCATE 1, 32
          PRINT FlnmP$;
        END IF
      LOOP WHILE (i = 0)
      IF (i <> 13) THEN
        LOCATE 1, 47
        PRINT a$;
        INPUT ; "", c$
        FlnmP$ = a$ + c$ + RIGHT$(Flnm$, 4)
      END IF
      PRINT
      CALL PLOT(-1, FlnmAdj$(FlnmP$, "E:\LAB\DAT\", "DAT"), psc)
        'Mode -1 -> Print frequency list
'      PRINT "Filename ="; FlnmP$
     CASE 114    'r
      iaux = iaux + 1
      IF (iaux > iauxmx) THEN iaux = 1
'        PRINT "Bias field (0.1 = 2mA) ["; AuxS$(iaux); "]      ";
'        LOCATE 27, 38
'        a$ = iptak$("")
'        ar = VAL(a$)
'        IF ((ar >= 0) AND (ar <= 5)) THEN AuxS$(iaux) = MID$(AuxS$(iaux), 1, 6) + STR$(ar)
     CASE 82     'R
      IF (iauxmx > 0) THEN
        l& = LEN(AuxS$(iaux))
        CALL koutputStr(0, nErr, SSEGADD&(AuxS$(iaux)), l&, 0, mAdAux(0), mTerm(0))
        IF nErr THEN PRINT "Aux. dev: IEEE Error #"; nErr
      END IF
     CASE 115    's
      PRINT "Intensity scaling for TOP beam of Pi/2-pulses (# or RET) ["; rtp!; "]:      ";
      LOCATE 27, 66
      a$ = iptak$("")
      ar = VAL(a$)
      IF ((ar > 0) AND (ar <= 1)) THEN
        rtp! = ar
        iMod = -1
      END IF
      LOCATE 27, 1
      PRINT "Intensity scaling for BOT beam of Pi/2-pulses (# or RET) ["; rbt!; "]:      ";
      LOCATE 27, 66
      a$ = iptak$("")
      ar = VAL(a$)
      IF ((ar > 0) AND (ar <= 1)) THEN
        rbt! = ar
        iMod = -1
      END IF
'     CASE 116    't
'      a$ = iptak$("Enter maximum number of auxillary device sequences(# or RET):")
'      ar = VAL(a$)
'      IF (a$ <> "") AND (ar >= 0) AND (ar <= 6) THEN
'        iauxmx = ar
'        iaux = 1
'      END IF
'     CASE 84     'T
'      PRINT "Enter auxillary device sequence #"; iaux; "string: ";
'      i = 1
'      b$ = ""
'      DO
'        a$ = getak$
'        IF (ASC(LEFT$(a$, 1)) <> 13) THEN
'          MID$(b$, i, 1) = a$
'          i = i + 1
'        END IF
'      LOOP WHILE (ASC(LEFT$(a$, 1)) <> 13)
'      IF b$ <> "" THEN AuxS$(iaux) = b$
     CASE 112    'p
      CLS '    *****     Menu for editing Raman pulse descriptions     *****
1200 LOCATE 1, 1
      PRINT "                          Pi Pulse Parameters           "
      PRINT " 1   Delay time before starting the sequence is "; usec$(CDBL(Tstart)); Clear$;
      PRINT " 2   Number of preselection pi pulses is"; nPiP; Clear$;
      PRINT " 3   Total time for a preselection pi pulse is "; usec$(iTpiP / fSamp#); Clear$;
      PRINT " 4   Number of main pi pulses is"; nPi; Clear$;
      PRINT " 5   Total time for a main pi pulse is "; usec$(iTpi / fSamp#); Clear$;
      PRINT " 6   Spacing of main pi pulses is "; usec$(iFFpi * Tff#); Clear$;
      PRINT " 7   Times spent on fast transitions are "; usec$(iTt1P / fSamp#); "/"; usec$(iTt1 / fSa
mp#); Clear$;
      PRINT " 8   Delays between turn-on and turn-off times are "; usec$(iTdP / fSamp#); "/"; usec$(i
Td / fSamp#); Clear$;
      PRINT " 9   Duration of frequency strobes is "; usec$((iTs - 1) / fSamp#); Clear$;
      PRINT " a   Delay between pulses and frequency strobes is "; usec$(iTdS / fSamp#); Clear$;
      PRINT " b   AWFG sample period is "; usec$(nCl / fCl); Clear$;
      PRINT " c   Rate parameters for pulse transitions are"; STR$(Cwf(0)); ","; Cwf(2); Clear$;
      PRINT " d   Equal intensity level is"; CINT(exp2 * xAmp); "% of full power"; Clear$;
      PRINT " e   Relative intensity of preselection pulses is"; CINT(exp2 * yP); "% of full power";
Clear$;
      PRINT "                         Pi/2 Pulse Parameters          "
      PRINT " f   Time between pi/2 pulses is "; msec$(iTint& / fSamp#); Clear$;
      PRINT " g   Total time for a pi/2 pulse is "; usec$(iTpiH / fSamp#); Clear$;
      PRINT " h   Time for fast transitions is "; usec$(iTt1H / fSamp#); "/"; usec$(iTtyH / fSamp#);
Clear$;
      PRINT " i   Delays between turn-on and turn-off times are "; usec$(iTdH / fSamp#); Clear$;
      PRINT " j   Time at equal intensities with common control on is "; usec$(iTon1 / fSamp#); Clear
$;
      PRINT " k   Time at equal intensities with individual controls on is "; usec$(iTon2 / fSamp#);
Clear$;
      PRINT " l   Rate parameters for pulse transitions are"; STR$(CwfH(0)); ","; CwfH(2); Clear$;
      PRINT " m   Equal intensity level is"; CINT(exp2 * xAmpH); "% of full power"
      PRINT " n   Phaselock settling time for pi/2's is "; usec$(iTpl / fSamp#); Clear$;
      PRINT " o   Minimum time for switching tracer OFF between pi/2's is "; usec$(CDBL(TbMin)); Clea
r$;
      PRINT " p   Mininum time for which gate is OFF between pi/2's is "; usec$(CDBL(TgMin)); Clear$;
      LOCATE 28, 1: PRINT Clear$;
      LOCATE 28, 1: PRINT "Change any (# or RET)?"
'      DO: a$ = INKEY$: LOOP WHILE a$ = ""
      a$ = getak$
      LOCATE 28, 1

      SELECT CASE ASC(a$)
       CASE 27   'ESC
        GOTO 1000
       CASE 13   'CR
        GOTO 1000
       CASE 49   '1
'        INPUT "Enter the delay time before starting the sequence (#us or RET):"; a$
        a$ = iptak$("Enter the delay time before starting the sequence (#us or RET): ")
        IF a$ <> "" THEN
          xSave = Tstart: Tstart = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN Tstart = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 50   '2
'        INPUT "Enter number of preselection pi pulses (# or RET):"; a$
        a$ = iptak$("Enter number of preselection pi pulses (# or RET): ")
        n = INT(VAL(a$))
        IF a$ <> "" AND n >= 0 THEN
          nSave = nPiP: nPiP = n: CALL Clock(-1)
          IF iErr THEN nPiP = nSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 51   '3
'        INPUT "Enter the total time for a preselection pi pulse (#us or RET):"; a$
        a$ = iptak$("Enter the total time for a preselection pi pulse (#us or RET): ")
        IF a$ <> "" THEN
          xSave = TpiP: TpiP = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN TpiP = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 52   '4
'        INPUT "Enter number of main pi pulses (# or RET):"; a$
        a$ = iptak$("Enter number of main pi pulses (# or RET): ")
        n = INT(VAL(a$))
        IF a$ <> "" AND n >= 0 THEN
          nSave = nPi: nPi = n: CALL Clock(-1)
          IF iErr THEN nPi = nSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 53   '5
'        INPUT "Enter the total time for a main pi pulse (#us or RET):"; a$
        a$ = iptak$("Enter the total time for a main pi pulse (#us or RET): ")
        IF a$ <> "" THEN
          xSave = Tpi: Tpi = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN Tpi = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 54   '6
'        INPUT "Enter the time between main pi pulses (#us or RET):"; a$
        a$ = iptak$("Enter the time between main pi pulses (#us or RET): ")
        IF a$ <> "" THEN
          xSave = Tsp: Tsp = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN Tsp = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 55   '7
'        INPUT "Enter fractional time spent on fast transitions (0-.5 or RET):"; a$
        a$ = iptak$("Enter fractional time spent on fast transitions (0-.5 or RET): ")
        ar = VAL(a$)
        IF a$ <> "" AND ar >= 0 AND ar <= .5 THEN
          xSave = xTt1: xTt1 = ar: CALL Clock(-1)
          IF iErr THEN xTt1 = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 56   '8
'        INPUT "Enter fractional delays between turn-on and turn-off (0-.5 or RET):"; a$
        a$ = iptak$("Enter fractional delays between turn-on and turn-off (0-.5 or RET): ")
        ar = VAL(a$)
        IF a$ <> "" AND ar >= 0 AND ar <= .5 THEN
          xSave = xTd: xTd = ar: CALL Clock(-1)
          IF iErr THEN xTd = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 57   '9
'        INPUT "Enter the duration of frequency strobes (#us or RET):"; a$
        a$ = iptak$("Enter the duration of frequency strobes (#us or RET): ")
        IF a$ <> "" THEN
          xSave = Ts: Ts = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN Ts = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 97   'a
'        INPUT "Enter the delay between pulses and frequency strobes (#us or RET):"; a$
        a$ = iptak$("Enter the delay between pulses and frequency strobes (#us or RET): ")
        IF a$ <> "" THEN
          xSave = TdS: TdS = VAL(a$) / exp6: CALL Clock(-1)
          IF iErr THEN TdS = xSave: CALL Clock(0) ELSE iMod = -1
        END IF
       CASE 98   'b
'        INPUT "Enter the minimum number of 200ns clock cycles per AWFG point (# or RET):"; a$
        a$ = iptak$("Enter the minimum number of 200ns clock cycles per AWFG point (# or RET): ")
        n = INT(VAL(a$))
```

```
      IF a$ <> "" AND n >= 1 THEN nClMin = n: CALL Clock(-1)
      iMod = -1 'Changing nClMin can't cause an error
     CASE 99   'c
'       INPUT "Enter the 1st rate parameter (>0 or RET):"; a$
      a$ = iptak$("Enter the 1st rate parameter (>0 or RET): ")
      IF a$ <> "" THEN
        Cwf(0) = VAL(a$): Cwf(1) = xAmp / (EXP(Cwf(0) / 2) - 1)
        iMod = -1
      END IF
      LOCATE 28, 1: PRINT Clear$; : LOCATE 28, 1
'       INPUT "Enter the 2nd rate parameter (>0 or RET):"; a$
      a$ = iptak$("Enter the 2nd rate parameter (>0 or RET): ")
      IF a$ <> "" THEN
        Cwf(2) = VAL(a$): Cwf(3) = (1 - xAmp) / (EXP(Cwf(2) / 2) - 1)
        iMod = -1
      END IF
     CASE 100  'd
'       INPUT "Enter the fractional power at equal intensity point (0-1 or RET):"; a$
      a$ = iptak$("Enter the fractional power at equal intensity point (0-1 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= 1 THEN
        xAmp = ar: Cwf(1) = xAmp / (EXP(Cwf(0) / 2) - 1)
        Cwf(3) = (1 - xAmp) / (EXP(Cwf(2) / 2) - 1)
        iMod = -1
      END IF
     CASE 101  'e
'       INPUT "Enter the fractional intensity for preselection pulses (0-1 or RET):"; a$
      a$ = iptak$("Enter the fractional intensity for preselection pulses (0-1 or RET): ")
      ar = VAL(a$): IF a$ <> "" AND ar > 0 AND ar <= 1 THEN yP = ar: iMod = -1
     CASE 102  'f
'       INPUT "Enter the time between pi/2 pulses (#ms or RET):"; a$
      a$ = iptak$("Enter the time between pi/2 pulses (#ms or RET): ")
      IF a$ <> "" THEN
        xSave = Tint: Tint = VAL(a$) / exp3
        LOCATE 28, 1
        q$ = UCASE$(iptak$("Should this value be rounded to the nearest missed recoil overlap (Y/n)
? "))
        IF q$ <> "N" THEN Tint = CLNG(fmr * Tint) / fmr
        CALL Clock(-1)
        IF iErr THEN Tint = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 103  'g
'       INPUT "Enter the total time for a pi/2 pulse (#us or RET):"; a$
      a$ = iptak$("Enter the total time for a pi/2 pulse (#us or RET): ")
      IF a$ <> "" THEN
        xSave = TpiH: TpiH = VAL(a$) / exp6: CALL Clock(-1)
        IF iErr THEN TpiH = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 104  'h
'       INPUT "Enter fractional time spent on fast transitions (0-.5 or RET):"; a$
      a$ = iptak$("Frac. time for phase sensitive fast transitions (0-.5 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= .5 THEN
        xSave = xTt1H: xTt1H = ar: CALL Clock(-1)
        IF iErr THEN xTt1H = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
      LOCATE 28, 1: PRINT Clear$; : LOCATE 28, 1
      a$ = iptak$("Frac. time for pure-state fast transitions (0-.5 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= .5 THEN
        xSave = yTt1H: yTt1H = ar: CALL Clock(-1)
        IF iErr THEN yTt1H = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 105  'i
'       INPUT "Enter fractional delays between turn-on and turn-off (0-.5 or RET):"; a$
      a$ = iptak$("Enter fractional delays between turn-on and turn-off (0-.5 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= .5 THEN
        xSave = xTdH: xTdH = ar: CALL Clock(-1)
        IF iErr THEN xTdH = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
```

```
     CASE 106  'j
'       INPUT "Enter fractional time with common control on (0-1 or RET):"; a$
      a$ = iptak$("Enter fractional time with common control on (0-1 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= 1 THEN
        xSave = xTon1: xTon1 = ar: CALL Clock(-1)
        IF iErr THEN xTon1 = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 107  'k
'       INPUT "Enter fractional time with individual controls on (0-1 or RET):"; a$
      a$ = iptak$("Enter fractional time with individual controls on (0-1 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= 1 THEN
        xSave = xTon2: xTon2 = ar: CALL Clock(-1)
        IF iErr THEN xTon2 = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 108  'l
'       INPUT "Enter the 1st rate parameter (>0 or RET):"; a$
      a$ = iptak$("Enter the 1st rate parameter (>0 or RET): ")
      IF a$ <> "" THEN
        CwfH(0) = VAL(a$): CwfH(1) = xAmpH / (EXP(CwfH(0) / 2) - 1)
        iModH = -1
      END IF
      LOCATE 28, 1: PRINT Clear$; : LOCATE 28, 1
'       INPUT "Enter the 2nd rate parameter (>0 or RET):"; a$
      a$ = iptak$("Enter the 2nd rate parameter (>0 or RET): ")
      IF a$ <> "" THEN
        CwfH(2) = VAL(a$): CwfH(3) = (1 - xAmpH) / (EXP(CwfH(2) / 2) - 1)
        iModH = -1
      END IF
     CASE 109  'm
'       INPUT "Enter the fractional power at equal intensity point (0-1 or RET):"; a$
      a$ = iptak$("Enter the fractional power at equal intensity point (0-1 or RET): ")
      ar = VAL(a$)
      IF a$ <> "" AND ar >= 0 AND ar <= 1 THEN
        xAmpH = ar: CwfH(1) = xAmpH / (EXP(CwfH(0) / 2) - 1)
        CwfH(3) = (1 - xAmpH) / (EXP(CwfH(2) / 2) - 1)
        iModH = -1
      END IF
     CASE 110  'n
'       INPUT "Enter the phaselock settling time (#us or RET):"; a$
      a$ = iptak$("Enter the phaselock settling time (#us or RET): ")
      IF a$ <> "" THEN
        xSave = Tpl: Tpl = VAL(a$) / exp6: CALL Clock(-1)
        IF iErr THEN Tpl = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 111  'o
'       INPUT "Enter the minimum time for which tracer is OFF (#us or RET):"; a$
      a$ = iptak$("Enter the minimum time for which tracer is OFF (#us or RET): ")
      IF a$ <> "" THEN
        xSave = TbMin: TbMin = VAL(a$) / exp6: CALL Clock(-1)
        IF iErr THEN TbMin = xSave: CALL Clock(0) ELSE iModH = -1
      END IF
     CASE 112  'p
'       INPUT "Enter the minimum time for which gate is OFF (#us or RET):"; a$
      a$ = iptak$("Enter the minimum time for which gate is OFF (#us or RET): ")
      IF a$ <> "" THEN
        xSave = TgMin: TgMin = VAL(a$) / exp6: CALL Clock(-1)
        IF iErr THEN TgMin = xSave: CALL Clock(0) ELSE iModH = -1
      END IF

    END SELECT
    GOTO 1200

  END SELECT
  GOTO 1005

END SUB
```

C.1.2 Data.BAS

Once all of the input parameters are chosen in the Menu module, the patterns are generated, and the external devices programmed, the *Data* module of the AltInt code actually acquires the data. It programs the analog-to-digital (AD) pc-board inside the computer to sample the photomultiplier tube (PMT) signal when the probe laser is gated on. It steps through the different frequencies and interferometer geometries of the scan, graphically displaying the data on the computer as it goes. And finally, when the scan ends, it saves the data and all of the input parameter settings to the computer's hard drive.

```
'Program DataXX.BAS
'      This module contains the Scan% function used to collect the A/D data.
'      It is modified from DATA41.BAS by rounding frequencies sent to the data
'      file to 0.1 mHz instead of 1 mHz and by transferring FSKDrive from
'      AltInt41 since the Cs lock tuning routines caused OUT OF MEMORY error.
'Revision History:
'43  2/25/97    Frequency offset values stored in sig(1..nSeq,1,1..nSteps)
'             modified to compensate for the phase shift from the 1.05us
'             delay when the DDS changes frequencies.
'50  7/23/97    Incorporate new direct digital synthesizer (ADS-431).
'                      fstep=0.00532.. --> fstep=0.432
'52 10/30/97    Same as version 50, except that:
'                 - ADS-431 synthesizer clocks at 1000 instead of 928 MHz
'                 - COMMON SHARED variable PfPi2&(m,i) stores the binary
'                   representation of the Pi/2-pulse center frequencies.
'                   The function Scan uses this variable to calculate
'                   cor#(m,n), the correction to the each frequency point
'                   due to the known phase error and fixed FSK delay of the
'                   ADS synthesizer
'                 - polarity of the REQ1 line of the AT-DIO-32F board is
'                   reversed (now active on RISING) because polarity of
'                   STROBE line to synthesizer, which is the same signal
'                   reversed
'53  2/12/98   Change method of scaling data so that the units are no
'             longer arbitrary.  In previous versions, the data taken by
'             the AtoD PCboard were not converted to volts and were
'             arbitrarily multiplied by scaling factors 'ysc' and 'yysc',
'             one for the main plot and one for the inset plot.  The data
'             are now plotted and stored in volts.  The scale factors define
'             the scale size for the plots.  Thus, ysc=0.1 implies that the
'             top of the screen is B+0.1Volt and the zero-line is B, where
'             B is the background value.  Similarly for yysc and the inset
'             plot.
'               This version now displays the vertical scale in the top center
'             of the screen and the offset in the lower right corner of each
'             subscreen.
'               The k0 parameter is decremented by one before saving it to the
'             data file, so that k0 is now defined to be the largest integer
'             such all of the data points when multiplied by 10^k0 fit on the
'             SCREEN with vertical scale ysc.
'54  4/ 2/98   Add code to acquire data from the tilt sensor on channels
'             2 and 3 of DAS-16 board.  Two tilt values (one for each channel)
'             are acquired for each valid data point (including background
'             points).  These values are averaged, converted to volts, and
'             stored in the output file.
'56  7/25/01   When the interferometer sequence was something atypical (i.e.
'             not 12, 34, or 1234, the correction for the ADS-431 synthesizer
'             was calculated wrong, because in two places I incorrectly indexed
'             frequencies with the sequence number 'j' instead of the
'             interferometer reference number 'm'.  After fixing this problem,
'             I checked the code by comparing the correction frequencies for
'             "atypical" sequences 1122 and 3344 with the "typical" sequence
'             1234 that we trust.
'
'*54:$INCLUDE: 'e:\lab\inc\nihpdas.inc'
'*54:$INCLUDE: 'e:\lab\inc\xmaw.inc'
'*55b
'$INCLUDE: 'inc\nihpdas.inc'
'$INCLUDE: 'inc\xmaw.inc'
'*55e

DEFINT I-N

DECLARE SUB basdasg (iMode, BYVAL dummy%, nErr)
DECLARE SUB BinCon (f#, fRndB&, fRnd#)
DECLARE FUNCTION Scan% (mSt(), sh$, df$(), bl$(), pr$(), fCAve#(), fBack&(), nFSK0, iUp)
DECLARE FUNCTION FSKdrive% (iMode, f#, lBuf&, FLckHDL&, nElem&)
DECLARE FUNCTION ADSPhase# (pf AS LONG)

'$INCLUDE: 'e:\lab\inc\altint.inc'
```

```
'$DYNAMIC

REM $STATIC
FUNCTION ADSPhase# (pf AS LONG)
'                                        JMH     10/30/97
'  Assume bit "b" (b=1..32, 32 is MSB) of the 32-bit frequency word "pf"
'incurs a delay of: T0 + (37-b)*Tclk, where T0 is some arbitrary fixed delay
'independent of "b" and Tclk = 1/Fclk is the clock period.  Ignorning the
'fixed delay T0, this implies that bit "b" produces a phase shift of
'(37-b)/2^(32-b) cycles.  This function returns the sum
'Sum[(37-B)/2^(32-B),{B}] where {B} is the set of all bits in "pf" which are
'ON

DIM B AS INTEGER
DIM phi AS DOUBLE
DIM ul AS LONG

  ul = pf
  phi = 0
  FOR B = 0 TO 30
    IF ((ul MOD 2) = 1) THEN phi = phi + (36 - B) / 2 ^ (31 - B)
    ul = FIX(ul / 2)
  NEXT B
  ADSPhase# = phi

END FUNCTION

FUNCTION FSKdrive% (op AS INTEGER, rf AS DOUBLE, lnBuf AS LONG, iHDL AS LONG, lnFSK AS LONG)
' Control routine for DDS synthesizer ver. 2.0  JMH  8/96
'   Since the frequency strobe for the DDS synthesizer is now controlled by
'   the AWFG board, that board is controlled via register commands to output
'   a pulse that latches in the center frequency for Reset mode.
'  Parameters:
'   op      - selects an operation:
'     0: configure AT-DIO-32F board, and set output frequency of DDS-1 to rf
'         - all input parameters, except rf, are ignored
'     1: arm AT-DIO-32F board for DMA transfers
'     2: disarm DMA transfers
'         - all input parameters ignored
'   rf      - frequency (Hz) to which the synthesizer is set upon reset
'   lnBuf   - size of frequency data buffer (number of 32-bit elements)
'   iHDL    - pointer to the frequency data buffer
'   lnFSK   - number of 32-bit entries in frequency data buffer
'              - must be smaller than lnBuf to allow for memory page alignment
'  Returns:
'   1, on success
'   0, on any error condition
'
DIM en AS INTEGER             'error code returned by NI-DAQ routines
DIM devcod AS INTEGER         'device code returned by Init.DA.Brds()
DIM aOff AS LONG              'page alignment offset returned by Align.DMA.Buffer

DIM i AS INTEGER
DIM ln AS LONG

'Register addresses for the AT-DIO-32F board:
CONST diodev% = 2     'device number of AT-DIO-32F board
CONST dioaddr% = 576  '(0x0240) base address of AT-DIO-32F board
CONST aCFG1% = dioaddr          'Configuration and Status Register Group (16-bit)
CONST aCFG2% = dioaddr + 2
CONST aCFG3% = dioaddr + 4
CONST aCFG4% = dioaddr + 20
CONST aPTA% = dioaddr + 6        'Digital I/O Port Register Group (8- or 16-bit)
CONST aPTB% = dioaddr + 7
CONST aPTC% = dioaddr + 8
CONST aPTD% = dioaddr + 9

'Register addresses for the PCIP-AWFG board:
CONST awfgaddr% = 800 '(0x0320) base IO address for the PCIP-AWFG board
CONST aXL% = awfgaddr     'x-channel data
CONST aXH% = awfgaddr + 1
```

```
CONST aYL% = awfgaddr + 2 'y-channel data
CONST aYH% = awfgaddr + 3
CONST aCON% = awfgaddr + 4 'Control/Status
CONST aDAC% = awfgaddr + 5 'Load DAC address
CONST aSYS% = awfgaddr + 6 'Load System address
CONST aSCL% = awfgaddr + 8 'End of Scan address
CONST aSCH% = awfgaddr + 9
CONST aCT0% = awfgaddr + 12   'Number of scans counter
CONST aCT1% = awfgaddr + 13   'Clock divider 1
CONST aCT2% = awfgaddr + 14   'Clock divider 2
CONST a8254% = awfgaddr + 15  '8254 Control/Status

'Miscellaneous constants
'CONST fstep = 5.827066091346741D-03    'Synthesizer step size (DDS-1)
CONST fstep = 1000000000# / 2147483648#  'Synthesizer step size (ADS-431)

CONST pl = 1      'level of diagnostic printing (0=none)

SELECT CASE op

CASE 0      'Reset

  IF pl > 1 THEN PRINT "Initializing AT-DIO-32F board for 32-bit DMA transfer."
  en% = Init.DA.Brds(diodev, devcod%)
  IF (en%) THEN
    PRINT "FSKdrive: Error after calling Init.DA.Brds: "; en%
    FSKdrive = 1
    EXIT FUNCTION
  END IF

  IF pl > 0 THEN PRINT "Setting DDS board to initial conditions:"
  IF pl > 0 THEN PRINT "  - Amplitude = 0.6 Vpp (MAX)"
  CFG1 = 0
  OUT aCFG1, CFG1: OUT aCFG1 + 1, INT(CFG1 / &H100)
  '0000 0000 0000 0000
  'Group 1 DMA disabled, Group 1 interrupt disabled, TDELAY1 = 0 ns,
  ' Port B handshaking disabled, Port A handshaking disabled, REQ1 active high,
  ' Port A not double-buffered, Group 1 level sensitive, ACK1 active high,
  ' ACK1 = 0, OUT1 = 0
  CFG2 = 0
  OUT aCFG2, CFG2: OUT aCFG2 + 1, INT(CFG2 / &H100)
  '0000 0000 0000 0000
  'Group 2 DMA disabled, Group 2 interrupt disabled, TDELAY2 = 0 ns,
  ' Port D handshaking disabled, Port C handshaking disabled, REQ2 active high,
  ' Port C not double-buffered, Group 2 level sensitive, ACK2 active high,
  ' ACK2 = 0, OUT2 = 0
  CFG3 = &H7F00
  OUT aCFG3, CFG3: OUT aCFG3 + 1, INT(CFG3 / &H100)
  '0111 1111 0000 0000
  'Port B not double-buffered, Port D writes, Port B writes, 32-bit transfer mode,
  ' Port C writes, Port A writes, Counter 2 counts Counter 3,
  ' Counter 1 counts Counter 3, no double DMA mode, Counter 3 interrupt disabled,
  ' Counter 2 disconnected, Counter 1 disconnected, Counter 2 disabled,
  ' Counter 1 disabled, Group 2 DMA terminal counts disabled,
  ' Group 1 DMA terminal counts disabled
  CFG4 = 0
  OUT aCFG4, CFG4: OUT aCFG4 + 1, INT(CFG4 / &H100)
  '0000 0000 0000 0000
  'Add TDELAY2 before ACK2 pulse, Add TDELAY1 before ACK1 pulse,
  ' Port D not double-buffered, Version C option disabled
  '
  'NOTE that the above configurations will be overwritten by the
  'National Instruments library routines called below.
  '
  IF pl > 0 THEN PRINT "  - Phase = 0"
  'Set and latch in phase = 0, latch in amplitude settings:
  OUT aPTA, 0: OUT aPTB, 0: OUT aPTC, 0: OUT aPTD, 0
  CFG1 = CFG1 OR 1
  OUT aCFG1, CFG1 'Set OUT1 high
  FOR i = 1 TO 1000: NEXT i   'Delay while filter capacitors charge
  CFG1 = CFG1 AND &HFE
```

```
  OUT aCFG1, CFG1 'Set OUT1 low

  'Write the frequency data
  IF (rf < 0) OR (rf > 1.25E+07) THEN
    ln = CLNG(1E+07 / fstep)         'Default to 10 MHz
    IF pl > 0 THEN PRINT "  - Frequency = 10,000,000.000 Hz"
  ELSE
    ln = CLNG(rf / fstep)
    IF pl > 0 THEN
      PRINT "  - Frequency = ";
      PRINT USING "#,###,###.###"; rf;
      PRINT " Hz"
    END IF
  END IF
  OUT aPTA, ln AND &HFF 'Write 4 lowest bits
  ln = INT(ln / &H100)
  OUT aPTB, ln AND &HFF
  ln = INT(ln / &H100)
  OUT aPTC, ln AND &HFF
  ln = INT(ln / &H100)
  OUT aPTD, ln AND &HFF
  FOR i = 1 TO 1000: NEXT i 'Wait while filter capacitors charge

  'Now use Channel X bit #0 of the AWFG to create a strobe pulse.
  ' When the program is compiled, the pulse is high for about 350us.
  ' First reset the AWFG board.
  iCON = 8 'LONG=0/2CH=0/200NS=0/RESET=0/XCLK=1/GATEN=0/NSCN=0/RUN=0
  OUT aCON, (iCON OR &H10) 'Reset
  OUT aCON, iCON 'Set control register
  FOR i = 0 TO 2 'Write data L->H->L
      'Reset system memory pointer to start of pattern
      OUT aSCL, 0: OUT aSCH, 0: OUT aDAC, 0: OUT aSYS, 0
      'Write 0, 1, 0
      OUT aXL, (i AND 1): OUT aXH, 0
      'Need to run in order to latch in new DAC value
      OUT aCON, (iCON OR 1) 'Run
      OUT aCON, iCON 'Stop
  NEXT i

  'Program AT-DIO-32F board:
  en% = DIG.Grp.Config(diodev, 1, 4, 0, 1)
  '32-bit transfer using single-buffered handshaking
  IF (en%) THEN
    PRINT "FSKdrive: Error after calling DIG.Grp.Config: "; en%
    FSKdrive = 1
    EXIT FUNCTION
  END IF
'  en% = DIG.Grp.Mode(deviceNumber,group,signal,edge,reqPol,ackPol,ackDelayTime)
'  en% = DIG.Grp.Mode(diodev, 1, 0, 0, 0, 0, 0)
  ' Leading-edge sensitive, REQ signals active on falling,
  ' ACK signals active on falling, ACK delay time = 0 ns
  en% = DIG.Grp.Mode(diodev, 1, 0, 0, 1, 0, 0)
  ' Leading-edge sensitive, REQ signals active on rising,
  ' ACK signals active on falling, ACK delay time = 0 ns
  IF (en%) THEN
    PRINT "FSKdrive: Error after calling DIG.Grp.Mode: "; en%
    FSKdrive = 1
    EXIT FUNCTION
  END IF
  en% = DIG.Block.PG.Config(diodev, 1, 2, 1, 4, 100, 0)
  ' Enable single-buffered output, REQ originates externally,
  '  [reqInterval timebase = 1ms], [reqInterval = 100 x timebase],
  '  external gate disabled
  IF (en%) THEN
    PRINT "FSKdrive: Error after calling DIG.Block.PG.Config: "; en%
    FSKdrive = 1
    EXIT FUNCTION
  END IF

CASE 1        'Arm for DMA transfer
```

```
 IF pl > 1 THEN PRINT "Page-aligning the memory buffer."
 en% = Align.DMA.Buffer(diodev, 13, iHDL, lnFSK, lnBuf, aOff)
 IF (en%) THEN
   PRINT "FSKdrive: Error after calling Align.DMA.Buffer: "; en%
   FSKdrive = 1
   EXIT FUNCTION
 END IF
 IF (aOff <> 0) THEN
   PRINT "FSKdrive: WARNING! Align.DMA.Buffer shifted FSK buffer to offset = "; aOff
 END IF
 IF pl > 1 THEN PRINT "Installing DMA driver."
 en% = DIG.Block.Out(diodev, 1, iHDL, lnFSK)
 IF (en%) THEN
   PRINT "FSKdrive: Error after calling DIG.Block.Out: "; en%
   FSKdrive = 1
   EXIT FUNCTION
 END IF

CASE 2

   en% = DIG.Block.Clear(diodev, 1)
      IF (en%) THEN
        PRINT "FSKdrive: Error after calling DIG.Block.Clear: "; en%
        FSKdrive = 1
        EXIT FUNCTION
      END IF
      FSKdrive = 0
      EXIT FUNCTION

  IF pl > 1 THEN PRINT "Pattern completed."

END SELECT

END FUNCTION

' Function Scan% ---
'     Collects the A/D data as the frequencies are scanned for an interfero-
'   meter or pi pulse sequence.  The data-taking cycle is as follows:
'
'    (1) Set detection pulse times
'    (2) Select FF, Cs detuning, and AWFG waveforms for next pulse sequence
'    (3) Update FSK array for next frequency
'    (4) Copy FSK array to DMA buffer
'    (5) Wait until end of FSK pattern
'    (6) Start detection at rising edge of IP0
'    (7) Collect PMT readings through A/D #0
'    (8) Process and plot new data
'    (9) Repeat (1)-(8) until end of scan, periodically taking background
'            readings
'   (10) Save data
'
FUNCTION Scan% (mSt(), sh$, df$(), bl$(), pr$(), fCAve#(), fBack&(), nFSK0, iUp)

'Register addresses for the PCIP-AWFG board:
CONST awfgaddr% = 800 '(0x0320) base IO address for the PCIP-AWFG board
CONST aXL% = awfgaddr     'x-channel data
CONST aXH% = awfgaddr + 1
CONST aYL% = awfgaddr + 2 'y-channel data
CONST aYH% = awfgaddr + 3
CONST aCON% = awfgaddr + 4 'Control/Status
CONST aDAC% = awfgaddr + 5 'Load DAC address
CONST aSYS% = awfgaddr + 6 'Load System address
CONST aSCL% = awfgaddr + 8 'End of Scan address
CONST aSCH% = awfgaddr + 9
CONST aCT0% = awfgaddr + 12   'Number of scans counter
CONST aCT1% = awfgaddr + 13   'Clock divider 1
CONST aCT2% = awfgaddr + 14   'Clock divider 2
CONST a8254% = awfgaddr + 15  '8254 Control/Status

'Miscellaneous constants
CONST exp2 = 100#, exp3 = 1000#
```

```
CONST smask = "##.##^^^^", fmask = ",########Hz"
CONST sigexp = " ##.#####^^^^", pcnt = "  _+_-####.###%"
CONST fFmt = "#########.####", tFmt = "####.###ms"
CONST fCent = 28000000# 'Frequency for the DF 0->0 transition
CONST nCyc = 2 'Number of data cycles before saving data
CONST nStat = 9 'Lines of statistics output to file (except for backgrounds)
CONST nXpts = 640, nYpts = 480 'Screen plot dimensions
CONST fCl = 5000000# 'AWFG maximum sample rate
CONST fstep = 1000000000# / 2147483648#  'Synthesizer step size (ADS-431)
CONST Tdds = 2.7E-08    'delay between FSK request and
                        ' actual frequency change (sec)
CONST diodev% = 2 'device number of AT-DIO-32F board
CONST ipl = 0 'Print error messages?

'$DYNAMIC
DIM sig(1 TO nSeq, 1 TO 2, nSteps) 'Sig (int#,freq/signal,pt#)
DIM cor#(1 TO nSeq, nSteps)      'correction to sig(int#,1,pt#)
DIM Bck4(1 TO nSeq, nBlk), AveBck4(1 TO nSeq) 'Bck(int#,bckgnd#)
DIM sMax(1 TO nSeq), sMin(1 TO nSeq) 'Extreme signals
DIM fMax(1 TO nSeq), fMin(1 TO nSeq) 'Location of extreme signals
DIM fs#(1) 'Temporary storage of rounded frequencies
DIM Ave(1 TO nSeq), sd(1 TO nSeq), frac(1 TO nSeq) 'Signal statistics
DIM dum, SCX, SCY, sum, sum2
DIM iAptr, FHDL%, FLckHDL&, AHDL%, ALckHDL&
DIM m, m2, iyC, ixC, nRptCnt, i, ii, j, jj, k, iBlk, NxtBlk
DIM iElem&, iDir, nErr, iPlot, fRndB&, iBck, iBckStp, ia, iX, nWait, iValid
DIM arb$, aFlnm AS STRING
DIM Tint#

DIM tltxs AS LONG, tltys AS LONG        'accumulators for tilt averages
  tltxs = 0: tltys = 0
  ntlt = 0                              'number of tilt samples acquired

  Tint# = nCl * iTint& / fCl

    PRINT "Initializing the AWFG board."
    'Determine the control register setting iCON and the clock divider iDiv
    IF nCl = 1 THEN
      'iCON = &H64 '2CH/200ns/EXT GATE
      iCON = &H66 '2CH/200ns/EXT GATE/NSCN
      'iCON = &H60 '2CH/200ns
      iDiv = 2 'Minimum setting without disabling 8254 outputs
    ELSE 'Don't set 200ns bit
      'iCON = &H44 '2CH/EXT GATE
      iCON = &H46 '2CH/EXT GATE/NSCN
      'iCON = &H40 '2CH
      iDiv = nCl
    END IF
    OUT aCON, iCON 'Set control register
    OUT a8254, &H74 'Program 8254 Counter #1 for internal timebase divider
    OUT aCT1, iDiv MOD &H100 'Low byte
    OUT aCT1, INT(iDiv / &H100) 'High byte
    'Calculate End of Scan Address
    iSCL = (lArb& - 1) MOD &H100: iSCH = INT((lArb& - 1) / &H100)

    PRINT "Selecting output waveforms."
    IF NOT iUp THEN 'Select AWFG pattern if not done within loop
      nErr = XMSSendBuff%(mSeq(1)) 'Copy waveform #j from XMS to AWFG board
      IF nErr THEN PRINT "XMSSendBuff: Error #"; nErr
    END IF
    'Feed Forward control
    outexpr = "FUNC:USER ARB_" + CHR$(48 + mSt(1)) + ";SHAP USER;:TRIG:SOUR EXT"
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth2(0), mTerm(0))
    IF nErr <> 0 THEN PRINT "FF Synth: Error #"; nErr
    'Cs detuning control
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
    IF nErr <> 0 THEN PRINT "Cs Synth: Error #"; nErr

    PRINT "Allocate memory for DMA transfers from A/D board."
    'Allocate an array of 16-bit (2 byte) data for storage of A/D data.
```

```
    '  This buffer must have 2*nSamp elements so it can be aligned to avoid
    '  page crossings.
    nByte = 2: lBuf& = 2 * nSamp: nElem& = nSamp: memType = 1: memSrce = 0
    nErr = NI.DAQ.Mem.Alloc%(AHDL%, nByte, lBuf&, memType, memSrce)
    IF nErr <> 0 THEN PRINT "NI.DAQ.Mem.Alloc%: Error #"; nErr
    nErr = NI.DAQ.Mem.Lock%(AHDL%, ALckHDL&)
    IF nErr <> 0 THEN PRINT "NI.DAQ.Mem.Lock%: Error #"; nErr
    'Find starting segment iAptr to avoid page crossing
    nErr = Align.DMA.Buffer(diodev, 11, ALckHDL&, nElem&, lBuf&, aOff&)
    IF nErr THEN PRINT "Align.DMA.Buffer: Error #"; nErr
    iAptr = ALckHDL& / &H10000 'Strip off segment from ALckHDL&
    iAptr = iAptr + INT((aOff& + .5) / 8) 'Shift 1 seg/16 bytes(8 elem)
    IF ipl THEN PRINT "AHDL="; AHDL%; "ALckHDL="; HEX$(ALckHDL&); "aOff="; HEX$(aOff&); "iAptr="; HE
X$(iAptr)

    PRINT "Allocate memory for DMA transfers to FSK synthesizer."
    'Allocate an array of 32-bit (4 byte) data for the FSK buffer.
    '  Each measurement, one of the four FSK patterns will be copied into
    '  this buffer.  This buffer must be twice as long as the pattern
    '  (2*nFSK elements) so it can be aligned to avoid page crossings for
    '  DMA transfers.
    nByte = 4: lBuf& = 2 * nFSK: nElem& = nFSK: memType = 1: memSrce = 0
    nErr = NI.DAQ.Mem.Alloc%(FHDL%, nByte, lBuf&, memType, memSrce)
    IF nErr <> 0 THEN PRINT "NI.DAQ.Mem.Alloc%: Error #"; nErr: STOP
    nErr = NI.DAQ.Mem.Lock%(FHDL%, FLckHDL&)
    IF nErr <> 0 THEN PRINT "NI.DAQ.Mem.Lock%: Error #"; nErr: STOP
    IF ipl THEN PRINT "FHDL="; FHDL%; "FLckHDL&="; HEX$(FLckHDL&)

    PRINT "Setting pulse times."
    m = mSt(1) 'First sequence
    'Set shutter, blasting, and probe times for SRS pulser#3
    outexpr = sh$ 'Always set the shutter time
    'Set the blasting time if not usually set
    IF NOT iUp OR iFix THEN outexpr = outexpr + bl$(m)
    'Set the probe time if not usually set
    IF NOT iUp THEN outexpr = outexpr + pr$(m)
    l& = LEN(outexpr)
    CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP2(0), mTerm(0))
    IF nErr <> 0 THEN PRINT "IEEE Error #"; nErr

    'Set DF Raman pulse time for SRS pulser#2
    'Set the DF Raman pulse time(s) if not usually set
    IF NOT iUp OR iFix THEN
      outexpr = df$(m): l& = LEN(outexpr)
      CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
      IF nErr <> 0 THEN PRINT "IEEE Error #"; nErr
    END IF

    'Prepare to start
    CLS : COLOR 7 'Light white for writing on plot

    LOCATE 28, CINT(41 - LEN(Flnm$) / 2): PRINT Flnm$;

    LOCATE 1, 38: PRINT USING "#^^^^V"; ysc;    'display main plot scale
    LOCATE 1, 59: PRINT USING "#^^^^V"; yysc;   'display inset plot scale
    'Draw plot borders and center lines
    IF iINT(0) THEN 'Single plot
      LOCATE 30, 1: PRINT "Tpr="; : PRINT USING tFmt; exp3 * Tpr(1);
      LINE (0, 384)-(639, 384), 15: LINE (320, 0)-(320, 479), 15
      SCX = (nXpts - 1) / nSteps: SCY = nYpts - 1
    ELSE 'Four plots
      FOR m = 1 TO 4
         IF iINT(m) THEN
            iyC = 15 * INT((m + 1) / 2): ixC = 46 - 40 * (m MOD 2)
            LOCATE iyC, ixC: PRINT "Tpr="; : PRINT USING tFmt; exp3 * Tpr(m);
         END IF
      NEXT m
      LINE (320, 0)-(320, 479), 9: LINE (0, 240)-(639, 240), 9
      LINE (0, 192)-(639, 192), 15: LINE (0, 432)-(639, 432), 15
      LINE (160, 0)-(160, 479), 15: LINE (480, 0)-(480, 479), 15
      SCX = (nXpts - 1) / (2 * nSteps): SCY = (nYpts - 1) / 2
```

```
    END IF
    'Statistics labels
    FOR j = 1 TO nSeq
      iyC = 15 * INT((mSt(j) + 1) / 2) - 13: ixC = 41 - 40 * (mSt(j) MOD 2)
      LOCATE iyC, ixC: PRINT "Sig=";
      LOCATE iyC + 1, ixC: PRINT "Max=";
      LOCATE iyC + 2, ixC: PRINT "Min="
    NEXT j
    LINE (0, 0)-(639, 479), 9, B 'Border box
    SCXi = .2 * (nXpts - 1) / nSamp: SCYi = .2 * (nYpts - 1) 'Inset scale factors

    iElem& = 0: iDir = 1 'Parameters for memory routines
    iBlk = 0 'Counter for background measurements
    nWait = nCyc 'Number of cycles before recording data
    iWait = 0 'Wait until user presses another key to record data?
    iQuit = 0 'Clear quit flag
    i = 0 'Data point
    nStart = 0: nStop = 0 'Counters for AWFG pattern status
    DO 'Make measurements at nSteps+1 pulse frequencies
      j = 1 'Sequence #
      DO 'Step through nSeq sequences
        iBckStp = 0 'Flag to tell that the program is not being backstepped
        Sig4 = 0 'Clear signal data
        nRptCnt = 1 'Repetition #
        DO 'Step through nRptC measurements for each point
            NxtBlk = iBlk * nSteps / (nBlk - 1) 'Value of i for next background
            iBck = (i = NxtBlk) 'Read background for this cycle?
            IF iBck THEN nRptC = 1 ELSE nRptC = nRpt 'Number of repetitions
            m = mSt(j) 'FF pattern and array storage pointer this cycle
            m2 = (m + 1) \ 2 'Which pair of interferometers?
            OUT aCON, iCON 'Clear START bit for AWFG board
            'Reset AWFG board address pointers and end-of-scan address
            OUT aSCL, 0: OUT aSCH, 0
            OUT aDAC, 0: OUT aSYS, 0
            OUT aSCL, iSCL: OUT aSCH, iSCH
            IF iUp THEN 'Update pulse times and waveforms for this cycle
               'Set blasting and probe time for SRS pulser#3
               outexpr = pr$(m)
               IF NOT iFix THEN outexpr = outexpr + ";" + bl$(m)
               l& = LEN(outexpr)
               CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP2(0), mTerm(0))
               IF nErr <> 0 THEN PRINT "Detection timing: Error #"; nErr
               IF NOT iFix THEN 'Set DF Raman pulse times for SRS pulser#2
                  outexpr = df$(m): l& = LEN(outexpr)
                  CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSRSP1(0), mTerm(0))
                  IF nErr <> 0 THEN PRINT "DF Raman timing: Error #"; nErr
               END IF
               'Copy AWFG waveform #mSeq(j) from XMS to AWFG
               nErr = XMSSendBuff%(mSeq(j))
               IF nErr THEN PRINT "XMSSendBuff: Error #"; nErr
               'Update the FF and Cs detuning waveforms
               outexpr = "FUNC:USER ARB_" + CHR$(48 + m) + ";SHAP USER"
            ELSE 'Rezero the phase for the FF and Cs detuning patterns
               'The HP 33120A Synthesizer improperly changes the burst phase
               ' after each trigger, so it must be rezeroed each cycle.
               outexpr = "BM:PHAS 0"
            END IF
            'Send command to FF and Cs detuning synthesizers
            l& = LEN(outexpr)
            CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth2(0), mTerm(0))
            IF nErr <> 0 THEN PRINT "FF synth: Error #"; nErr
            CALL koutputStr(0, nErr, SSEGADD&(outexpr), l&, 0, mAdSnth3(0), mTerm(0))
            IF nErr <> 0 THEN PRINT "Cs synth: Error #"; nErr
            OUT a8254, &H30 'Program 8254 Counter #0 for 1 cycle
            OUT aCT0, 1: OUT aCT0, 0 'Low byte, high byte
            OUT aCON, iCON OR 1 'Set START bit for AWFG pattern

            'Update scanned FSK frequencies
            IF nFSK0 = 0 THEN 'Scan preselection frequency
               'Since the preselection frequency is loaded onto the DDS lines
               ' by the last strobe of the previous cycle, we must determine
```

```
               ' the frequency for the next cycle.
               iCyc = (nRptCnt = nRpt AND nWait = 0) 'End of cycle?
               iNxt = i - (iCyc AND NOT iBck)
               iBckNxt = ((iNxt = NxtBlk) AND NOT (iCyc AND iBck))
               IF iBckNxt THEN 'Background next point
                  fRndB& = fBack&(m, 0)
               ELSE 'Calculate frequency for normal data
                  CALL BinCon(fC#(m, 0) + fOff - fspan * (.5 - iNxt / nSteps), fRndB&, fs#(0))
               END IF
               FSK&(2, m) = fRndB& 'Store frequency at end of FSK array
               IF NOT iBckNxt AND iNxt <= nSteps THEN
                  'Store actual frequency - center offset
                  sig(j, 1, iNxt) = fs#(0) - fCAve#(m)
                  cor#(j, iNxt) = 0     'no correction for Pi-pulse scans
               END IF
            ELSE 'Do not scan preselection frequency
               FOR k = 0 TO 1 + iINT(0) 'May need to scan 2 frequencies
                  IF iBck THEN 'Background this point
                     fRndB& = fBack&(m, k)
                  ELSE 'Calculate frequency for normal data
                     CALL BinCon(fC#(m, k) + fOff - fspan * (.5 - i / nSteps), fRndB&, fs#(k))
                  END IF
                  FSK&(nFSK0 + k, m) = fRndB&
               NEXT k
               IF NOT iBck THEN 'Store actual frequency - center offset
                  sig(j, 1, i) = (fs#(0) + fs#(1 + iINT(0))) / 2 - fCAve#(m)
'*55                  cor#(j, i) = (PfPi2&(j, 1) - PfPi2&(j, 2)) * fstep * Tdds / Tint#
'*56:
                  cor#(j, i) = (PfPi2&(m, 1) - PfPi2&(m, 2)) * fstep * Tdds / Tint#
                    'compensate for FSK delay between 1st and 2nd Pi/2-pulses
                  cor#(j, i) = cor#(j, i) + (fs#(0) - fs#(1 + iINT(0))) * Tdds / Tint#
                    'compensate for FSK delay between 3rd and 4th Pi/2-pulses
'*55                  cor#(j, i) = cor#(j, i) + (ADSPhase(PfPi2&(j, 1)) - ADSPhase(PfPi2&(j, 2))) /
Tint#
'*56:
                  cor#(j, i) = cor#(j, i) + (ADSPhase(PfPi2&(m, 1)) - ADSPhase(PfPi2&(m, 2))) / Tint
#
                    'compensate for phase error of ADS-431 going from 1st to 2nd Pi/2
                  cor#(j, i) = cor#(j, i) + (ADSPhase(FSK&(nFSK0, m)) - ADSPhase(FSK&(nFSK0 + 1, m))
) / Tint#
                    'compensate for phase error of ADS-431 going from 3rd to 4th Pi/2
               END IF
            END IF

            'Copy FSK sequence into buffer for DMA transfer
            '  Skip 1st element FSK&(0,m) to rotate pattern
            nErr = NI.DAQ.Mem.Copy%(FHDL%, FSK&(1, m), iElem&, nElem&, iDir)
            IF nErr <> 0 THEN PRINT "Error in NI.DAQ.Mem.Copy #"; nErr

            'Arm Digital I/O board for DMA transfers to synthesizer.
            ' Routine will immediately return control to main program
            nErr = FSKdrive%(1, fCent / 4, lBuf&, FLckHDL&, nElem&)

            'Wait until IP0 is low for correct probe triggering
            DO: CALL basdasg(14, VARPTR(iParlst(0)), nErr)
            LOOP WHILE (iParlst(0) AND 1) = 1

            'Set up to start aquisition on rising trigger IP0
            'The call routine is not terminated until IP0 goes high
            '# conversions; Memory segment for DMA; 1=trigger from timer; 0 = One shot; gain set
            iParlst(0) = nSamp: iParlst(1) = iAptr: iParlst(2) = 1: iParlst(3) = 0: iParlst(4) = 0
            CALL basdasg(6, VARPTR(iParlst(0)), nErr) 'Mode 6
            IF nErr <> 0 THEN PRINT "BASDASG mode 6: Error #"; nErr

            IF nWait = 0 THEN 'Check whether AWFG pattern was completed
               IF (INP(aCON) AND 1) THEN
                  LOCATE 14, 1: PRINT "i="; i; "j="; j; ": AWFG not completed"
                  nWait = 1 'Wait 1 cycle for valid pattern
                  nStart = nStart + 1
               ELSE
                  nStop = nStop + 1
```

```
               END IF
            END IF

            'Wait until conversion complete
            DO   'check status mode (8)
            CALL basdasg(8, VARPTR(iParlst(0)), nErr)
            LOOP WHILE (iParlst(1) AND 1) = 1

            'Disarm DMA transfers for synthesizer
            ' Now that the probe has ended, it is safe to disarm the DMA
            ' transfers for the synthesizer, and check whether the AWFG
            ' pattern ended its cycle properly.
            nErr = FSKdrive%(2, fCent / 4, lBuf&, FLckHDL&, nElem&)

            'Retrieve probe A/D data
            ' Transfer from segment iAptr; Conversion 0; Transfer to iData(); No channel array
            iParlst(0) = nSamp: iParlst(1) = iAptr: iParlst(2) = 0: iParlst(3) = VARPTR(iData(0)): i
Parlst(4) = 0
            CALL basdasg(9, VARPTR(iParlst(0)), nErr) 'Mode 9
            IF nErr <> 0 THEN PRINT "BASDASG Mode 9: Error #"; nErr

'Acquire tilt signals:
    'Set the DAS-16 channel limits using mode 1 to sample channel 2 and 3
    iParlst(0) = 2: iParlst(1) = 3 'lower limit; upper limit
    CALL basdasg(1, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 1: Error #"; nErr: iQuit = -1
    'Sample once from channel 2
    CALL basdasg(3, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 3: Error #"; nErr: iQuit = -1
    itltx = iParlst(0)
    'Sample once from channel 3
    CALL basdasg(3, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 3: Error #"; nErr: iQuit = -1
    itlty = iParlst(0)
'restore DAS-16 configuration
    'Set the DAS-16 channel limits using mode 1 to sample channel 0
    iParlst(0) = 0: iParlst(1) = 0 'lower limit; upper limit
    CALL basdasg(1, VARPTR(iParlst(0)), nErr)
    IF nErr <> 0 THEN PRINT "BASDASG mode 1: Error #"; nErr: iQuit = -1

            'Plot and sum the A/D samples
            LINE (nXpts, 0)-(nXpts * .8, nYpts * .2), 1, BF 'Draw inset box
            sum = 0
            FOR k = 0 TO nSamp - 1
               y = iData(k) / 4095 * 10
               sum = sum + y
               y = .9 * (1 - (y - Bck4(j, 0)) / yysc)
               IF (y >= 0) AND (y < 1) THEN
                 PSET (CINT(.8 * nXpts + SCXi * k), CINT(SCYi * y)), 15
               END IF
            NEXT k
            sum = sum / nSamp
            'Display background value
            IF iINT(0) THEN
              iyC = 29: ixC = 71
            ELSE
              iyC = 1 + 14 * INT((m + 1) / 2): ixC = 71 - 40 * (m MOD 2)
            END IF
            LOCATE iyC, ixC: PRINT USING "##.#^^^^V"; Bck4(j, 0);

            'Check for any keypress
            a$ = UCASE$(INKEY$)
            IF a$ <> "" THEN
               IF a$ <> "Q" AND a$ <> "B" THEN iWait = NOT iWait 'Toggle iWait
               'Check for Q(uit) by user
               IF a$ = "Q" THEN
                  iQuit = -1: iAuto = 0 'Return to menu
               END IF
               'Check for B(ack step) by user.  If so, back up in frequency.
               IF a$ = "B" THEN 'Back up the signal and bckgnd pointers
                  iPlot = i + (j = 1) 'Number of last plotted point
```

```
                LOCATE 5, 2: PRINT "Point #"; STR$(iPlot);
                INPUT ": Back step amount ="; a$: ia = VAL(a$)
                LOCATE 5, 2: PRINT STRING$(37, 32);
                iBckStp = (ia > 0) 'Only backstep if positive input
                IF iBckStp THEN
                   i = iPlot - ia 'i + 1 is value of i for next measurement
                   IF i < -1 THEN i = -1
                   DO WHILE ((iBlk - 1) * nSteps / (nBlk - 1)) > i
                      iBlk = iBlk - 1
                   LOOP
                   iBck = 0 'Don't let iBlk advance
                   'Redetermine the extrema in case they've been erased
                   FOR jj = 1 TO nSeq
                      FOR ii = 0 TO i
                         dum = sig(jj, 2, ii)
                         IF dum > sMax(jj) OR ii = 0 THEN sMax(jj) = dum: fMax(jj) = sig(jj, 1, ii
)
                         IF dum < sMin(jj) OR ii = 0 THEN sMin(jj) = dum: fMin(jj) = sig(jj, 1, ii
)
                      NEXT ii
                   NEXT jj
                END IF
                nWait = 2 'Wait through one cycle to get back in synch
             END IF
          END IF
          IF iWait THEN nWait = 2 'Don't record data until keypress
          iValid = (nWait = 0) 'Data invalid if exit before nWait=0
          IF iValid THEN
             Sig4 = Sig4 + sum 'Include data in sum
             nRptCnt = nRptCnt + 1 'Increment nRptCnt
          ELSE
             nWait = nWait - 1 'Decrement nWait
          END IF
       LOOP WHILE (nRptCnt <= nRptC) AND NOT (iQuit OR iBckStp)

       IF iValid THEN 'Check for minimum or maximum and plot
    'average acquired tilt values:
    ntlt = ntlt + 1
    tltxs = tltxs + itltx
    tltys = tltys + itlty
'    LOCATE 14, 10
'    PRINT "X:"; itltx; ", "; itltx / 4095 * 10; ", "; tltxs / 4095 * 10 / ntlt; ", "; ntlt
'    LOCATE 15, 10
'    PRINT "Y:"; itlty; ", "; itlty / 4095 * 10; ", "; tltys / 4095 * 10 / ntlt; ", "; ntlt
'          Sig4 = ysc * Sig4 / nRptC
         Sig4 = Sig4 / nRptC
         iX = CINT(SCX * (i + ((m - 1) AND 1) * nSteps))
         IF iBck THEN 'Save and plot background value
           Bck4(j, iBlk) = Sig4
           'Subtract background (Use 1st bckgnd pt, average bckgnd's later)
           dum = Sig4 - Bck4(j, 0)
           'Make room for circles
           IF iX < 3 THEN iX = 3
           IF iX > nXpts - 3 THEN iX = nXpts - 3
           dum = dum / ysc
           FOR k = 0 TO 3 'Plot points on screen if valid
              IF dum > -.25 AND dum < 1 THEN
                 CIRCLE (iX, SCY * (m2 - .2 - .8 * dum)), 2, 10 + k
              END IF
              dum = dum * 10
           NEXT k
         ELSE 'Save and plot signal value
           sig(j, 2, i) = Sig4
           'Subtract background
           dum = Sig4 - Bck4(j, 0)
           'Print present, high, and low signals
           IF dum > sMax(j) OR i = 0 THEN sMax(j) = dum: fMax(j) = sig(j, 1, i)
           IF dum < sMin(j) OR i = 0 THEN sMin(j) = dum: fMin(j) = sig(j, 1, i)
           iyC = 15 * m2 - 13: ixC = 45 - 40 * (m MOD 2)
           LOCATE iyC, ixC: PRINT USING smask; dum; : PRINT USING fmask; sig(j, 1, i);
           LOCATE iyC + 1, ixC: PRINT USING smask; sMax(j); : PRINT USING fmask; fMax(j)
```

```
           LOCATE iyC + 2, ixC: PRINT USING smask; sMin(j); : PRINT USING fmask; fMin(j)
           'Plot data points over four orders of magnitude
           dum = dum / ysc / 10
           FOR k = 0 TO 3
              IF dum > -.25 AND dum < 1 THEN
                 PSET (iX, SCY * (m2 - .2 - .8 * dum)), 10 + k
              END IF
              dum = dum * 10
           NEXT k
         END IF
       END IF
       j = j + 1
     LOOP WHILE (j <= nSeq) AND iValid
     iBlk = iBlk - iBck 'Advance background pointer
     i = i - (NOT iBck) 'Advance point counter except after backgrounds
   LOOP WHILE (i <= nSteps) AND NOT iQuit

   'Set AWFG running continuously
   iCON = iCON AND &HFD 'Set NSCN=0 (-> continuous)
   OUT aCON, iCON 'Stop AWFG and set NSCN=0
   'Reset AWFG board address pointers and end-of-scan address
   OUT aSCL, 0: OUT aSCH, 0
   OUT aDAC, 0: OUT aSYS, 0
   'For cw operation, the last gate sample must cause the AWFG pattern to
   ' wrap back around to sample #0. (For single shot, setting START=1 did
   ' this.  Consequently, for a fixed gate pattern, one point must be
   ' dropped from the AWFG pattern. Since the first and last point are both
   ' at 0, it doesn't hurt to eliminate the last point.
   iSCL = (lArb& - 2) MOD &H100: iSCH = INT((lArb& - 2) / &H100)
   OUT aSCL, iSCL: OUT aSCH, iSCH
   OUT a8254, &H30 'Program 8254 Counter #0 for 1 cycle
   OUT aCT0, 1: OUT aCT0, 0 'Low byte, high byte
   OUT aCON, iCON OR 1 'Set START bit for AWFG pattern
   IF ipl THEN LOCATE 15, 1: PRINT "AWFG stopped:"; nStop; ", AWFG running:"; nStart

   IF iAuto = 1 THEN iQuit = -1 'Don't save data

   'Unlock and free memory for DMA transfers
   nErr = NI.DAQ.Mem.Unlock%(AHDL%)
   IF nErr THEN PRINT "NI.DAQ.Mem.Unlock for A/D: Error #"; nErr
   nErr = NI.DAQ.Mem.Free%(AHDL%)
   IF nErr THEN PRINT "NI.DAQ.Mem.Free for A/D: Error #"; nErr
   nErr = NI.DAQ.Mem.Unlock%(FHDL%)
   IF nErr THEN PRINT "NI.DAQ.Mem.Unlock for FSK: Error #"; nErr
   nErr = NI.DAQ.Mem.Free%(FHDL%)
   IF nErr THEN PRINT "NI.DAQ.Mem.Free for FSK: Error #"; nErr

   IF NOT iQuit THEN 'Save the data to the hard disk
     k0 = 3 'Maximum scale for plotting
     FOR j = 1 TO nSeq
        'Calculate average offset over all background readings
        AveBck4(j) = 0
        FOR i = 0 TO nBlk - 1
           AveBck4(j) = AveBck4(j) + Bck4(j, i)
        NEXT i
        AveBck4(j) = AveBck4(j) / nBlk

        'Process raw signal and generate statistics with average offset
        sum = 0: sum2 = 0
        FOR i = 0 TO nSteps
           'Scale and subtract offsets
           sig(j, 2, i) = sig(j, 2, i) - AveBck4(j)
           sum = sum + sig(j, 2, i)
           sum2 = sum2 + sig(j, 2, i) ^ 2
        NEXT i
        Ave(j) = sum / (nSteps + 1) 'Average signal
        sd(j) = SQR((sum2 - (nSteps + 1) * Ave(j) ^ 2) / nSteps) 'Standard deviation of signal
        IF Ave(j) <> 0 THEN
           frac(j) = ABS(exp2 * sd(j) / Ave(j)) 'Fractional rms noise in the signal
        ELSE
           frac(j) = 0
```

```
        END IF

        'Find maximum scale for plotting
        DO WHILE 10 ^ k0 * sMin(j) < -.1 OR 10 ^ k0 * sMax(j) > 1
           k0 = k0 - 1
        LOOP
     NEXT j
     k0 = k0 - 1

     'Open the file for storing data
     OPEN "e:\lab\dat\" + Flnm$ FOR OUTPUT AS 1

     'Save number of steps and number of sequences
     PRINT #1, nSteps; d; nSeq

     'Save frequency/signal pairs
     FOR k = 0 TO nSteps
        FOR j = 1 TO nSeq
           PRINT #1, USING fFmt; sig(j, 1, k);
           PRINT #1, USING sigexp; sig(j, 2, k);
           PRINT #1, USING fFmt; cor#(j, k);
        NEXT j
        PRINT #1,
     NEXT k

     'Save numerical parameters
     'Parameters for fit program only
     PRINT #1, nPi
     PRINT #1, USING "##.#########"; Tint#
     'Parameters for fit and Plot40 programs
     FOR j = 1 TO nSeq
        PRINT #1, mSeq(j)
        PRINT #1, USING fFmt; fCAve#(mSt(j))
     NEXT j
     'Save graphing parameters and # of lines of statistics for Plot??
'     PRINT #1, ysc; d; k0; d; nStat + nBlk
     PRINT #1, ysc; d; k0; d; nStat + nBlk + 2
       'two lines added for tilt values
     'Save summary statistics
     ' Print interferometer sequence as column headings
     PRINT #1, "Interferometer #:";
     FOR j = 1 TO nSeq: PRINT #1, mSeq(j); STRING$(10, 32); : NEXT j
     FOR i = 0 TO nBlk - 1
        IF i = 0 THEN
           PRINT #1, d; "F=4 Backgnd:";
        ELSE
           PRINT #1, d; "            ";
        END IF
        FOR j = 1 TO nSeq
           PRINT #1, USING sigexp; Bck4(j, i);
        NEXT j
     NEXT i
     PRINT #1, d; "Ave Backgnd:";
     FOR j = 1 TO nSeq
        PRINT #1, USING sigexp; AveBck4(j);
     NEXT j
     PRINT #1, d; "Max. Signal:";
     FOR j = 1 TO nSeq
        PRINT #1, USING sigexp; sMax(j);
     NEXT j
     PRINT #1, d; "@ fMax (Hz):";
     FOR j = 1 TO nSeq
        PRINT #1, USING fFmt; fMax(j);
     NEXT j
     PRINT #1, d; "Min. Signal:";
     FOR j = 1 TO nSeq
        PRINT #1, USING sigexp; sMin(j);
     NEXT j
     PRINT #1, d; "@ fMin (Hz):";
     FOR j = 1 TO nSeq
        PRINT #1, USING fFmt; fMin(j);
```

```
     NEXT j
     PRINT #1, d; "Ave. Signal:";
     FOR j = 1 TO nSeq
        PRINT #1, USING sigexp; Ave(j);
     NEXT j
     PRINT #1, d; "Stand. Dev.:";
     FOR j = 1 TO nSeq
        PRINT #1, USING sigexp; sd(j);
     NEXT j
     PRINT #1, d; "Fract. Err.:";
     FOR j = 1 TO nSeq
        PRINT #1, USING pcnt; frac(j);
     NEXT j
     PRINT #1, d; "X-tilt (V):";
     PRINT #1, USING " ##.###"; tltxs / 4095 * 10 / ntlt
     PRINT #1, "Y-tilt (V):";
     PRINT #1, USING " ##.###"; tltys / 4095 * 10 / ntlt
   END IF

   Scan% = iQuit

END FUNCTION
```

C.1.3 PlotFit.BAS

The *PlotFit* module is used to graphically display and fit the data. The fit code is a subset of the main fit code Fit.C given in section C.2.

```
'PFXX.BAS -- This module contains the Plot subroutine for outputting data
'           to the HP Plotter and the nonlinear least squares fit
'           routines for fitting interferometer data to sinusoids.
'            It is modified from PF42.BAS by checking the frequency data for
'           the formatted length based on the location of the first space.
'           This allows it to be compatible with different frequency formats,
'           which is necessary since Data42 rounds frequencies to 0.1 mHz
'           instead of 1 mHz.
'Revision History:
'43            Ring bell when scan is finished.
'52 11/ 5/97   Modify SUBROUTINE Plot() so that it can the correction to the
'             frequency values.  As soon as the frequency values freq(i,j) are
'             loaded from the data file, the correction is added to freq(i,j).
'             Thus, the graphing, plotting, and fitting, sections all use the
'             corrected frequencies.
'53  2/12/98   Plots are now scaled in the same way that DATA53 scales the
'             incoming data.  As in DATA53, the vertical scale is displayed
'             in the top center of the screen, and the offset is displayed in
'             the lower right of each sub-screen.  The section for ploting to
'             the HP7090A plotter has been removed and replaced with a section
'             which gives the user the option of changing the vertical scale.
'             The vertical scale is initial set to yscP/10^k0, where yscP and
'             k0 are values stored in the data file by DATA??.  k0 is the
'             largest integer such that all data points multiplied by 10^k0 fit
'             on the screen with the original vertical scale yscP.
'54  4/16/98   Add display of recoil difference in ppb and mrad.  Change
'             default answer to question "Fit this data?" from YES to NO.
'55  7/13/98 Changed fitting algorithm from Levenberg-Marquardt to linear
'             least squares.  Changed default answer to question "Fit this data?"
'             back from NO to YES.
'             --> Changed fit from sin(kx+p) to cos(kx+p)
'                 (now all phases are pi/2 less than they would be before)
'55 11/ 6/98   Plot() no longer calls FlnmAdj$() to add path and extension.
'             Thus it assumes Flnm$ is the complete filename.  Display the
'             file name when plotting.  Instead of querying, accept 'f' to run
'             fit.
'61  2/16/01   Ignore the graph scaling factor (or graph maximum value) yscP
'             saved in the data file.   Calculate the "best" scaling factor
'             by finding the maximum of all of the data and then rounding up
'             to the nearest value 1e-X,2e-X,5e-X, where X is the smallest
'             exponent possible
'
'*55 '$INCLUDE: 'e:\HP\BAS\head2.bi'

DEFINT I-N
DEFSNG A-H, O-Z

DECLARE SUB Plot (mode, Flnm$, psc)
DECLARE SUB Fit2 (Tint, nPts, nSeq, of(), amp(), idisc(), parm(), psd(), chi(), nErrFit())
DECLARE FUNCTION lfit (N, x(), y(), parm(), sig(), chisq)
DECLARE SUB standardform (j, p(), psd())
DECLARE FUNCTION mod2pi (phi)
DECLARE FUNCTION GaussJ% (N, A(), B())
'*60b
DECLARE SUB FitGhead (nPts, of(), amp(), idisc(), parm(), psd(), chi(), nErrFit())
DECLARE SUB FitG (nPts, xd(), yd(), ibad(), parm(), psd(), chi, nErrFit)
DECLARE SUB covsrt (nFit, ia(), covar())
DECLARE SUB mrqcof (nData, x(), y(), nFit, A(), ia(), alpha(), da(), chisq)
DECLARE FUNCTION mrqmin (nData, x(), y(), nFit, A(), ia(), covar(), alpha(), da(), chisq, aLamda)
DECLARE SUB funcs (x, A(), y, dyda())
DECLARE SUB PUNice (x!, xsd!, xnm$, xunit$)
'*60e

CONST nSeqMax = 4 'Maximum number of interferometer sequences
CONST pi = 3.141592654#, twopi = 2 * pi
CONST ERRTooFewPts = 1, ERRBadData = 2, ERRAllocMem = 3
CONST ERRNRAllocMem = 4, ERRNRSingMat = 5

'Constants for fit routines
CONST pl% = 1    ' level of diagnostic printing (0=least)
CONST nParm = 4           ' maximum number of parmeters in fit
```

```
CONST DontVary = 0, DoVary = -1, AutoVary = 1

CONST nSamp = 200 'Number of 20 us A/D samples for probe detection

'$STATIC
'variables for A/D board
DIM iParlst(15) 'A/D Driver requires iParLst() be Static
DIM iData(nSamp) 'Storage for A/D data
'variables for plot subroutine
DIM mTerm(3) 'GPIB terminators

'Common variables for A/D board
COMMON SHARED iParlst(), iData()
'Common variables for Plot subroutines
COMMON SHARED mTerm(), outexpr AS STRING, iErr

'$DYNAMIC

99    iErr = -1
      CLOSE 1
      PRINT "Plot: File I/O Error. Press any key to continue."
      DO: LOOP WHILE INKEY$ = ""
      RESUME NEXT

999   iErr = -1
      CLOSE 1
      RESUME NEXT

      END

REM $STATIC
DEFINT I-N

SUB covsrt (nFit, ia(), covar())
' Expand compressed covar(nFit,nFit) back into original parameter space
'   covar(nParm,nParm).
'
' (C) Copr. 1986-92 Numerical Recipes Software #Q2Zr!$!-3.
'

DIM i, j, k

   'Clear outer elements of covar()
   FOR i = nFit + 1 TO nParm
      FOR j = 1 TO i: covar(i, j) = 0: covar(j, i) = 0: NEXT j
   NEXT i

   'Expand covar() back into original parameter space
   k = nFit
   FOR j = nParm TO 1 STEP -1
      IF ia(j) THEN
         FOR i = 1 TO nParm: SWAP covar(i, k), covar(i, j): NEXT i
         FOR i = 1 TO nParm: SWAP covar(k, i), covar(j, i): NEXT i
         k = k - 1
      END IF
   NEXT j

'ERASE I, j, k

END SUB 'covsrt

SUB Fit2 (Tint, nPts, nSeq, of(), amp(), idisc(), parm(), psd(), chi(), nErrFit())
' Fit2 is a replacement for the Levenberg-Marquardt algorithm sub called Fit.
' This procedure calls the linear least squares fit of the data, throws away
' 3 sigma points, and then refits the data.
' Input parameters are
  'Tint' is the time between pi/2 pulses.  1/Tint is the fringe spacing.
  'nPts' is the number of data points
  'nSeq' is the number of interferometer sequences to fit
  'of'   is the array(0..nPts-1,1..nSeq) of offset frequencies
  'amp'  is the array(0..nPts-1,1..nSeq) of amplitudes
```

```
'Output parameters are
  'parm' is the array(0..nPts-1,1..nSeq) of fit parameters
  'psd'  is the array(1..nParm,1..nSeq) of fit parameters error bars
  'chi' is the array(1..nSeq) of chi^2 values returned by fit
'are output parameters.  Fit returns via nErrFit() integer(s) corresponding
'to one of the possible return codes defined in this file (search for "const err").

'$DYNAMIC

CONST DIS3SIGMA = 1   ' nonzero to discard points > 3sigma

DIM x(1 TO nPts)          ' scaled frequencies
DIM y(1 TO nPts)          ' scaled amplitudes
DIM z(1 TO nPts)          ' y_fit - y_data
DIM z1, z2                ' 1st and 2nd moments of z
DIM sigma                 ' RMS deviation of z ( = sqrt(z2 - z1*z1))
DIM nDiscard AS INTEGER
DIM fitparm(1 TO 4)
DIM fitpsd(1 TO 4)
DIM char AS INTEGER
DIM i

  FOR j = 1 TO nSeq 'Step through interferometer sequences

    FOR i = 0 TO nPts - 1
         x(i + 1) = twopi * Tint * of(i, j)'Scale frequency data
         y(i + 1) = amp(i, j)
    NEXT i

    IF nPts <= nParm + 1 THEN nErrFit(j) = ERRTooFewPts: GOTO Fit2End

    ' Get a fit using all the points
    nErrFit(j) = lfit(nPts, x(), y(), fitparm(), fitpsd(), chisq)
    IF nErrFit(j) GOTO Fit2End

    IF DIS3SIGMA = 0 THEN

      nDiscard = 0
      FOR i = 1 TO nPts
        idisc(i - 1, j) = 0
      NEXT i

    ELSE

      ' We will throw away the > 3sig points, so now
      ' do some statistics on the residual error

      z1 = 0
      z2 = 0
      FOR i = 1 TO nPts
        z(i) = fitparm(1) + fitparm(2) * COS(x(i) + fitparm(3)) - y(i)
        z1 = z1 + z(i)
        z2 = z2 + z(i) * z(i)
      NEXT i
      z1 = z1 / nPts
      z2 = z2 / nPts
      sigma = SQR(z2 - z1 * z1)

      ' Recopy x and y arrays, skipping points which are > 3 sigma
      nDiscard = 0
      place = 1
      FOR look = 1 TO nPts
        IF ABS(z(look) / sigma) > 3 THEN
          idisc(look - 1, j) = 1        ' -1 because discard is 0 to npts-1
          nDiscard = nDiscard + 1
        ELSE
          idisc(look - 1, j) = 0
          x(place) = x(look)
          y(place) = y(look)
          place = place + 1
        END IF
```

```
      NEXT look
'      PRINT nDiscard; " "; place; " "; look

      FOR i = 1 TO nPts - nDiscard
        z(i) = fitparm(1) + fitparm(2) * COS(x(i) + fitparm(3)) - y(i)
'         PRINT INT(ABS(z(I) / sigma)); " ";
      NEXT i
'      PRINT

      IF nDiscard > 0 THEN
'         PRINT "refitting, "; nPts - nDiscard; " points"
        IF nPts - nDiscard <= nParm + 1 THEN nErrFit(j) = ERRTooFewPts: GOTO Fit2End

        ' Get a fit using remaining points
        nErrFit(j) = lfit(nPts - nDiscard, x(), y(), fitparm(), fitpsd(), chisq)
        IF nErrFit(j) GOTO Fit2End
      END IF
    END IF

    parm(1, j) = fitparm(1)
    parm(2, j) = fitparm(2)
    parm(3, j) = 1                    ' we are not fitting the fringe spacing
    parm(4, j) = fitparm(3)

    psd(1, j) = fitpsd(1)
    psd(2, j) = fitpsd(2)
    psd(3, j) = 0
    psd(4, j) = fitpsd(3)

    chi(j) = chisq

    'Return to original scale

    CALL standardform(j, parm(), psd())

    IF pl% >= 10 THEN
      PRINT "signal atoms/standard deviation="; parm(2, j) / ysig
      PRINT "Press any key to continue."
      DO: LOOP WHILE INKEY$ = ""
      CLS
    END IF

    nErrFit(j) = 0

Fit2End: NEXT j

END SUB 'Fit2

SUB FitG (nPts, xd(), yd(), ibad(), parm(), psd(), chi, nErrFit)
'  Run the non-linear least-squared fit routines that use the
'Levenberg-Marquardt Method code from Numerical Recipes to fit to a
'gaussian: y(x) = p1 * Exp(-0.5*[(x-p2)/p3]^2) + p4
'Input parameters:
'  nPts        number of data points
'  xd, yd      arrays 1..Npts of the data to be fit
'Output parameters:
'  parm, psd   arrays 1..nParm fit parameters, error bars
'  chi         chi^2 value returned by fit
'  ibad        array 1..Npts of:
'                -1 if the point is discarded for the final fit
'                 0 otherwise
'  nErrFit     return error codes
'
'
'
'   Fit first finds the mean and standard deviation of the amplitude
'values in 'amp()'.  It then scales the data by dividing 'amp()' by its
'mean and multiplying 'of()' by 2*pi*Tint which converts 'of()' to radians.
'It leaves out any "drop-out" points that are more than 3 standard deviations
'away from the mean.  The scaled data are stored in 'x()' and 'y()'.
'   The fit function is given by: y = A + B sin(C x + D).  The fit
```

```
'routines thus vary the 4 parameters A,B,C,D to minimize chi^2, where chi^2
'is the sum of (y(i)-amp(i,j))^2.  The initial values of these parameters
''parminit()' are set using the mean and standard deviation as follows:
'  A = mean/range          DoVary
'  B = 1.3 * sdev/range    DoVary
'  C = 1.0                 DontVary
'  D = crossing of y=mean  DoVary
'The control array 'pvinit()' determines how each parameter is varied by the
'fit routine.  It has 3 possible values: DontVary, DoVary, AutoVary.
'"DontVary" tells the fit routine NOT to vary that parameter, while "DoVary"
'allows the parameter to vary.  "AutoVary" is used for extremely sensitive
'parameters.  "AutoVary" forces the fit to be run twice.  Any parameters set
'to "AutoVary" are held constant on the first pass and then allowed to vary
'on the second pass.  The final values of the parameters from the first pass
'are used as initial values for the second pass.  Fit returns the final
'parameter values from the best fit (i.e. the one that produced the smaller
'chi^2).
'
CONST head$ = "  i   amplitude   Foffset     width     Yoffset      chi^2       lamda"
CONST fmti$ = "###:", fmtA$ = " ##.###^^^^"
CONST fmtch$ = "  ##.###^^^^", fmtl$ = "######.####"

CONST DIS3SIGMA = 1   ' nonzero to discard points > 3sigma

'$DYNAMIC
DIM parminit(1 TO nParm)  ' initial values of fit parameters
DIM fitparm(1 TO nParm)   ' fit parameters that change each fit iteration
DIM dp(1 TO nParm)        ' parameter adjustment vector
DIM covar(1 TO nParm, 1 TO nParm) ' covariance matrix
DIM alpha(1 TO nParm, 1 TO nParm) ' curvature matrix
DIM x(1 TO nPts)          ' scaled frequencies
DIM y(1 TO nPts)          ' scaled amplitudes
DIM ymin, ymax, yrng      ' range of y-data
DIM xoff, xrng            ' offset, range for x-data
DIM z(1 TO nPts)          ' y_fit - y_data
DIM z1, z2                ' 1st and 2nd moments of z
DIM sigma                 ' RMS deviation of z ( = sqrt(z2 - z1*z1))
DIM y1, y2, ytar
DIM aLamda

DIM nData              ' number of data points used in fit
DIM nFit               ' number of parameters that actually vary in fit
DIM n2Pass              ' number of parameters set to AutoVary
DIM pvinit(1 TO nParm) AS INTEGER ' Handle parameter? DontVary,DoVary,AutoVary
DIM pvary(1 TO nParm) AS INTEGER  ' Vary this parameter? DontVary,DoVary
DIM iter               ' current number of fit iterations
DIM char AS INTEGER
DIM i, i1, i2, imax, nErr

   'Set parameter control variables
   pvinit(1) = DoVary: pvinit(2) = DoVary: pvinit(3) = DoVary: pvinit(4) = DoVary

   'Choose initial guesses for fit parameters
    IF pl% >= 20 THEN PRINT "Choosing initial values for fit parameters."
    xmax = xd(1): xmin = xd(1)
    ymax = yd(1): ymin = yd(1)
    imax = 1
    FOR i = 2 TO nPts       ' find extrema
      IF xd(i) < xmin THEN xmin = xd(i)
      IF xd(i) > xmax THEN xmax = xd(i)
      IF yd(i) < ymin THEN ymin = yd(i)
      IF yd(i) > ymax THEN
        ymax = yd(i)
        imax = i
      END IF
    NEXT i
    xoff = xd(imax)
    xrng = (xmax - xmin) / 2
    yrng = ymax - ymin
    ytar = ymin + yrng * EXP(-.5)
    y1 = yrng
```

```
    FOR i = 2 TO imax                   ' find Exp(-0.5) points
      y2 = ABS(yd(i) - ytar)
      IF (y2 < y1) THEN
        i1 = i
        y1 = y2
      END IF
    NEXT i
    y1 = yrng
    FOR i = imax + 1 TO nPts
      y2 = ABS(yd(i) - ytar)
      IF (y2 < y1) THEN
        i2 = i
        y1 = y2
      END IF
    NEXT i
    IF pl% >= 20 THEN PRINT "  xrange ="; xrng; " yrange ="; yrng;

    IF pl% >= 20 THEN PRINT "Scaling data."
    nData = 0 'Pointer for valid data array
    FOR i = 1 TO nPts
         nData = nData + 1
         x(nData) = (xd(i) - xoff) / xrng
         y(nData) = (yd(i) - ymin) / yrng
    NEXT i
    IF nData <= nParm + 1 THEN nErrFit = ERRTooFewPts: GOTO FitGexit
    'Set initial parameters
    parminit(1) = 1!
    parminit(2) = 0!
    parminit(3) = ABS(xd(i1) - xd(i2)) / 2 / xrng
    parminit(4) = 0!

    nFit = 0 '# of parameters to vary in the fit
    n2Pass = 0 '# of additional parameters to vary in 2nd pass
    FOR i = 1 TO nParm
      fitparm(i) = parminit(i)
      IF pvinit(i) = AutoVary THEN
         pvary(i) = DontVary
         n2Pass = n2Pass + 1
      ELSE
         pvary(i) = pvinit(i)
         nFit = nFit - (pvary(i) = DoVary)
      END IF
    NEXT i

    FOR iPass = 0 TO -(n2Pass > 0) '2 passes if n2Pass>0
      IF iPass THEN '2nd pass
         FOR i = 1 TO nParm 'Allow more parameters to vary on 2nd pass
            IF pvinit(i) = AutoVary THEN pvary(i) = DoVary
         NEXT i
         nFit = nFit + n2Pass
         IF pl% >= 10 THEN
            PRINT "Press any key to fit again for 'AutoVary' parameters:"
            DO: LOOP WHILE INKEY$ = ""
            CLS
         END IF
      END IF

      IF pl% > 0 THEN LOCATE 1, 1: PRINT head$
      chisq = 0: aLamda = -1!
      iter = 0 'Iteration counter
      DO
         IF pl% > 0 THEN
            IF pl < 10 THEN LOCATE 2, 1
            PRINT USING fmti$; iter;
            PRINT USING fmtA$; fitparm(1) * yrng;
            PRINT USING fmtA$; fitparm(2) * xrng + xoff;
            PRINT USING fmtA$; fitparm(3) * xrng;
            PRINT USING fmtA$; fitparm(4) * yrng + ymin;
            PRINT USING fmtch$; chisq / (nData - nFit) * yrng ^ 2;
            PRINT USING fmtl$; aLamda
            IF pl% >= 10 AND CSRLIN >= 27 THEN
```

```
               PRINT "Press any key to continue."
               DO: LOOP WHILE INKEY$ = ""
               CLS : PRINT head$
            END IF
         END IF
         nErrFit = mrqmin(nData, x(), y(), nFit, fitparm(), pvary(), covar(), alpha(), dp(), chisq,
aLamda)
         IF nErrFit GOTO FitGexit
         iter = iter + 1
      LOOP WHILE iter <= 100 AND aLamda <= 1000
      aLamda = 0!
      nErrFit = mrqmin(nData, x(), y(), nFit, fitparm(), pvary(), covar(), alpha(), dp(), chisq, aLa
mda)
      IF nErrFit GOTO FitGexit
      chisq = chisq / (nData - nFit)
      IF (NOT iPass) OR chisq <= chi THEN
         FOR i = 1 TO nParm 'Transfer parameters to return arrays
            parm(i) = fitparm(i)
            psd(i) = SQR(chisq * covar(i, i))
         NEXT i
         chi = chisq
         IF iPass AND pl% >= 10 THEN
            PRINT "New chi-squared better (smaller) than previous value."
            PRINT " -> use NEW parameters"
         END IF
      ELSE
         IF pl% >= 10 THEN
            PRINT "New chi-squared worse (larger) than prxevious value."
            PRINT " -> keep OLD parameters"
         END IF
      END IF
    NEXT iPass

  IF DIS3SIGMA = 0 THEN

      nDiscard = 0
      FOR i = 1 TO nPts
        ibad(i) = 0
      NEXT i

  ELSE

      ' We will throw away the > 3sig points, so now
      ' do some statistics on the residual error

      z1 = 0
      z2 = 0
      FOR i = 1 TO nPts
        z(i) = parm(1) * EXP(-.5 * ((x(i) - parm(2)) / parm(3)) ^ 2) + parm(4) - y(i)
        z1 = z1 + z(i)
        z2 = z2 + z(i) * z(i)
      NEXT i
      z1 = z1 / nPts
'      z2 = z2 / Npts
      sigma = SQR(z2 / (nPts - 1) - z1 * z1)

      ' Recopy x and y arrays, skipping points which are > 3 sigma
      nDiscard = 0
      place = 1
      FOR look = 1 TO nPts
        IF ABS(z(look) / sigma) > 3 THEN
          ibad(look) = 1
          nDiscard = nDiscard + 1
        ELSE
          ibad(look) = 0
          x(place) = x(look)
          y(place) = y(look)
          place = place + 1
        END IF
      NEXT look
'      PRINT nDiscard; " "; place; " "; look
```

```
    IF nDiscard > 0 THEN
      IF pl% > 0 THEN
        LOCATE 29, 2
        PRINT nDiscard; "points discarded.  Refitting"; nPts - nDiscard; "points";
        LOCATE 1, 1
        PRINT head$
      END IF
      IF nPts - nDiscard <= nParm + 1 THEN nErrFit(j) = ERRTooFewPts: GOTO Fit2End
      chisq = 0: aLamda = -1!
      iter = 0 'Iteration counter
      DO
         IF pl% > 0 THEN
            IF pl < 10 THEN LOCATE 2, 1
            PRINT USING fmti$; iter;
            PRINT USING fmtA$; fitparm(1) * yrng;
            PRINT USING fmtA$; fitparm(2) * xrng + xoff;
            PRINT USING fmtA$; fitparm(3) * xrng;
            PRINT USING fmtA$; fitparm(4) * yrng + ymin;
            PRINT USING fmtch$; chisq / (nData - nFit) * yrng ^ 2;
            PRINT USING fmtl$; aLamda
            IF pl% >= 10 AND CSRLIN >= 27 THEN
                PRINT "Press any key to continue."
                DO: LOOP WHILE INKEY$ = ""
                CLS : PRINT head$
            END IF
         END IF
         nErrFit = mrqmin(nData - nDiscard, x(), y(), nFit, fitparm(), pvary(), covar(), alpha(), dp
(), chisq, aLamda)
         IF nErrFit GOTO FitGexit
         iter = iter + 1
      LOOP WHILE iter <= 100 AND aLamda <= 1000
      aLamda = 0!
      nErrFit = mrqmin(nData, x(), y(), nFit, fitparm(), pvary(), covar(), alpha(), dp(), chisq, aLa
mda)
      IF nErrFit GOTO FitGexit
      chisq = chisq / (nData - nFit)
    END IF
  END IF

    'Return paramters to original scale
    parm(1) = yrng * parm(1)
    psd(1) = yrng * psd(1)
    parm(2) = xrng * parm(2) + xoff
    psd(2) = xrng * psd(2)
    parm(3) = xrng * parm(3)
    psd(3) = xrng * psd(3)
    parm(4) = yrng * parm(4) + ymin
    psd(4) = yrng * parm(4)
    chi = yrng ^ 2 * chi
    IF pl% >= 10 THEN
      PRINT "Press any key to continue."
      DO: LOOP WHILE INKEY$ = ""
      CLS
    END IF
    IF pl% > 0 THEN
      LOCATE 1, 1
      PRINT STRING$(75, 32)
      PRINT STRING$(75, 32)
      LOCATE 29, 1
      PRINT STRING$(75, 32);
    END IF

    nErrFit = 0

FitGexit:

ERASE parminit, fitparm, dp, covar, alpha, x, y, pvinit, pvary
'ERASE ymin, ymax, yrng, ymean, ysig, aLamda
'ERASE nData, nFit, n2Pass, iter, char, I, nErr
```

```
END SUB 'FitG

SUB FitGhead (nPts, of(), amp(), idisc(), parm(), psd(), chi(), nErrFit())
'   Header routine to interface non-linear least-squares fit code with the
'Plot() subroutine of PF??.BAS.
'Input parameters:
'  nPts         number of data points
'  of,
'   amp         arrays 0..Npts-1,1..nSeqMax of the data to be fit
'Output parameters:
'  parm, psd    arrays 1..nParm,1..nSeqMax of fit parameters, error bars
'  chi          array 1..nSeqMax of chi^2 values returned by fit
'  idisc        is the array 1..Npts,1..nSeqMax of:
'                 -1 if the point is discarded for the final fit
'                  0 otherwise
'  nErrFit      array 1..nSeqMax of return error codes
'
'$DYNAMIC
DIM xd(1 TO nPts)        ' x-data
DIM yd(1 TO nPts)        ' y-data
DIM ibad(1 TO nPts)
DIM fprm(1 TO nParm)     ' fit parameters
DIM fpsd(1 TO nParm)     ' fit parameters error bars
DIM chisq                ' chi-square value from fit
DIM nec                  ' fit return code

DIM i

  FOR i = 1 TO nPts
    xd(i) = of(i - 1, 1)
    yd(i) = amp(i - 1, 1)
  NEXT i

  CALL FitG(nPts, xd(), yd(), ibad(), fprm(), fpsd(), chisq, nec)

  FOR i = 1 TO nPts
    idisc(i - 1, 1) = ibad(i)
  NEXT i
  FOR i = 1 TO nParm
    parm(i, 1) = fprm(i)
    psd(i, 1) = fpsd(i)
  NEXT i
  chi(1) = chisq
  nErrFit(1) = nec

ERASE xd, yd, ibad, fprm, fpsd

END SUB 'FitGhead

REM $STATIC
'Evaluate the fit function and its derivatives:
'         y = p1 * Exp{-0.5*[(x-p2)/p3]^2} + p4
'    dy/dp1 = Exp{-0.5*[(x-p2)/p3]^2}
'    dy/dp2 = p1 * Exp{-0.5*[(x-p2)/p3]^2} * (x-p2) / p3^2
'    dy/dp3 = p1 * Exp{-0.5*[(x-p2)/p3]^2} * (x-p2)^2 / p3^3
'    dy/dp4 = 1
'
SUB funcs (x, p(), y, dydp())

DIM xs

   xs = (x - p(2)) / p(3)
   dydp(4) = 1
   dydp(1) = EXP(-(xs ^ 2) / 2)
   y = p(1) * dydp(1)
   dydp(2) = y * xs / p(3)
   dydp(3) = dydp(2) * xs
   y = y + p(4)

'ERASE xs
```

```
END SUB 'funcs

REM $DYNAMIC
'**********************************************************************
'Gauss-Jordan elimination routine from Numerical Recipes,
' modified to return error codes.
'
' (C) Copr. 1986-92 Numerical Recipes Software #Q2Zr!$!-3.
'**********************************************************************
'
FUNCTION GaussJ% (N, A(), B())
'
'$DYNAMIC

DIM big, dum, pivinv

DIM indxc(1 TO N), indxr(1 TO N), iPiv(1 TO N)
DIM i, iCol, iRow, j, k, l, ll

   FOR i = 1 TO N 'Loop over columns to be reduced
      big = 0!
      FOR j = 1 TO N 'Outer loop of search for pivot elements
         IF iPiv(j) <> 1 THEN
            FOR k = 1 TO N
               IF iPiv(k) = 0 THEN
                  IF ABS(A(j, k)) >= big THEN
                     big = ABS(A(j, k))
                     iRow = j
                     iCol = k
                  END IF
               ELSE
                  IF iPiv(k) > 1 THEN GaussJ% = ERRNRSingMat: GOTO GJexit
               END IF
            NEXT k
         END IF
      NEXT j
      iPiv(iCol) = iPiv(iCol) + 1
      IF iRow <> iCol THEN
         FOR l = 1 TO N: SWAP A(iRow, l), A(iCol, l): NEXT l
         SWAP B(iRow), B(iCol)
      END IF
      indxr(i) = iRow
      indxc(i) = iCol
      'Divide pivot row by pivot element
      IF (A(iCol, iCol) = 0!) THEN GaussJ% = ERRNRSingMat: GOTO GJexit
      pivinv = 1! / A(iCol, iCol)
      A(iCol, iCol) = 1!
      FOR l = 1 TO N: A(iCol, l) = pivinv * A(iCol, l): NEXT l
      B(iCol) = pivinv * B(iCol)
      'Reduce the rows
      FOR ll = 1 TO N
        IF ll <> iCol THEN
           dum = A(ll, iCol)
           A(ll, iCol) = 0!
           FOR l = 1 TO N: A(ll, l) = A(ll, l) - A(iCol, l) * dum: NEXT l
           B(ll) = B(ll) - B(iCol) * dum
        END IF
      NEXT ll
   NEXT i
   'Unscramble the column interchanges
   FOR l = N TO 1 STEP -1
      IF indxr(l) <> indxc(l) THEN
         FOR k = 1 TO N: SWAP A(k, indxr(l)), A(k, indxc(l)): NEXT k
      END IF
   NEXT l

   GaussJ% = 0

GJexit:
'ERASE big, dum, pivinv, I, iCol, iRow, j, k, l, ll
ERASE indxc, indxr, iPiv
```

```
END FUNCTION ’GaussJ

REM $STATIC
FUNCTION lfit (N, x(), y(), prm(), psd(), chisq)

’ Fit the data to prm(1) + prm(2) * (cos(x + prm(3)))
’ This is done by doing a linear least squares fit to the function
’ myparm(1) + myparm(2) sin(x) + myparm(3) cos(x)
’ and then calculating parm() from myparm().
’ The linear least squares algorithm comes from Numerical Recipes
’ The transfer over the the new variables is scribed from sinefit.c

  DIM design(1 TO N, 1 TO 3) AS SINGLE

  FOR i = 1 TO N                ’ the design matrix has elements composed of
    design(i, 1) = 1             ’  the basis functions located at the x
    design(i, 2) = COS(x(i))    ’  coordinates of the data points
    design(i, 3) = SIN(x(i))
  NEXT i

  DIM A(1 TO 3, 1 TO 3)         ’ these are alpha and beta in Numerical Recipes
  DIM B(1 TO 3)

  FOR i = 1 TO 3
    FOR j = 1 TO i
      A(i, j) = 0               ’ initialize to 0
      FOR k = 1 TO N
        A(i, j) = A(i, j) + design(k, i) * design(k, j)
      NEXT k
      A(j, i) = A(i, j)         ’ a is symmetric
    NEXT j
  NEXT i

  FOR i = 1 TO 3
    B(i) = 0
    FOR k = 1 TO N
      B(i) = B(i) + y(k) * design(k, i)
    NEXT k
  NEXT i

  ’ now solve the "normal equations" by calling gaussJ

  result = GaussJ(3, A(), B())
  IF result <> 0 THEN lfit = result: EXIT FUNCTION

  ’ calculate chisq

  chisq = 0
  FOR i = 1 TO N
    yfit! = B(1) + B(2) * COS(x(i)) + B(3) * SIN(x(i))
    diff! = yfit! - y(i)
    chisq = chisq + diff! * diff!
  NEXT i
  chisq = chisq / (N - 3)       ’ chisq /= ndata - nfit

  ’ calculate amplitude, contrast, phase, and uncertainties

  prm(1) = B(1)
  psd(1) = SQR(chisq * A(1, 1))
  f1 = B(2) * B(2) + B(3) * B(3)
  prm(2) = SQR(f1)
  psd(2) = SQR(chisq / f1 * (B(2) * B(2) * A(2, 2) + B(3) * B(3) * A(3, 3) + 2 * B(2) * B(3) * A(2,
3)))

  ’ phase is defined as cos(x + phase)
  ’ and should be atan2(-b(3),b(2))
  ’ except there is no atan2 in BASIC

  ’ the following code should give the right result and be in the range [-pi,pi]
  prm(3) = ATN(-B(3) / ABS(B(2)))
```

```
  IF (B(2) < 0) THEN prm(3) = 3.141592653589# - prm(3)
  IF (prm(3) > pi) THEN prm(3) = prm(3) - twopi

  psd(3) = SQR(chisq * (B(3) * B(3) * A(2, 2) + B(2) * B(2) * A(3, 3) - 2 * B(2) * B(3) * A(2, 3)))
/ f1

  lfit = 0

END FUNCTION ’lFit

’ Shift the angle phi by 2*pi steps so that it lies between +-pi.
’   mod2pi returns the integer number of steps required.
’
FUNCTION mod2pi (phi)

  j = INT((ABS(phi) + pi) / twopi)
  IF phi > 0 THEN j = -j
  phi = phi + j * twopi

  mod2pi = j

END FUNCTION ’mod2pi

REM $DYNAMIC
SUB mrqcof (nData, x(), y(), nFit, A(), ia(), alpha(), beta(), chisq)

’$DYNAMIC

DIM ymod, wt, dy, dyda(1 TO nParm)
DIM i, j, k, l, m

   FOR j = 1 TO nFit ’Initialize (symmetric) alpha() and beta()
      FOR k = 1 TO j: alpha(j, k) = 0!: NEXT k
      beta(j) = 0!
   NEXT j

   chisq = 0!
   FOR i = 1 TO nData ’Sum over all data to get alpha(), beta(), and chisq
      CALL funcs(x(i), A(), ymod, dyda())
      dy = y(i) - ymod
      j = 0
      FOR l = 1 TO nParm
         IF ia(l) THEN
            wt = dyda(l)
            j = j + 1
            k = 0
            FOR m = 1 TO l
               IF ia(m) THEN
                  k = k + 1
                  alpha(j, k) = alpha(j, k) + wt * dyda(m)
               END IF
            NEXT m
            beta(j) = beta(j) + dy * wt
         END IF
      NEXT l
      chisq = chisq + dy ^ 2
   NEXT i
   FOR j = 2 TO nFit ’Fill the symmetric side of alpha()
      FOR k = 1 TO j - 1
         alpha(k, j) = alpha(j, k)
      NEXT k
   NEXT j

’ERASE ymod, wt, dy, dyda, I, j, k, l, m
ERASE dyda

END SUB ’mrqcof

REM $STATIC
’*************************************************************************
’ Nonlinear least-squares fit routine from Numerical Recipes, modified to
```

```
’  return error codes.
’
’ (C) Copr. 1986-92 Numerical Recipes Software #Q2Zr!$!-3.
’*************************************************************************
’
FUNCTION mrqmin (nData, x(), y(), nFit, A(), ia(), covar(), alpha(), da(), chisq, aLamda)

’$DYNAMIC
DIM atry(1 TO nParm)                ’Trial parameter values
DIM beta(1 TO nFit)                 ’Parameter adjustment vector
DIM ochisq                          ’Storage of old chisq value
DIM j, k, l, m, nErr

   IF aLamda < 0 THEN ’Set initial values on 1st iteration
      aLamda = .001
      CALL mrqcof(nData, x(), y(), nFit, A(), ia(), alpha(), da(), chisq)
   END IF

   ’Augment diagonal elements of nonlinear fitting matrix
   ochisq = chisq ’Save old chisq value
   FOR j = 1 TO nFit
      FOR k = 1 TO nFit: covar(j, k) = alpha(j, k): NEXT k
      covar(j, j) = alpha(j, j) * (1! + aLamda)
      beta(j) = da(j)
   NEXT j

   ’Find the matrix solution
   nErr = GaussJ%(nFit, covar(), beta())
   IF nErr THEN mrqmin = nErr: GOTO Mrqexit

   IF aLamda = 0 THEN ’Expand covar() back into original parameter space
      CALL covsrt(nFit, ia(), covar())
      mrqmin = 0
      GOTO Mrqexit
   END IF

   ’Does chisq improve?
   j = 0
   FOR l = 1 TO nParm
      IF ia(l) THEN
         j = j + 1: atry(l) = A(l) + beta(j)
      ELSE
         atry(l) = A(l)
      END IF
   NEXT l
   CALL mrqcof(nData, x(), y(), nFit, atry(), ia(), covar(), beta(), chisq)

   IF chisq < ochisq THEN ’Accept new solution
      aLamda = .1 * aLamda
      FOR j = 1 TO nFit
         FOR k = 1 TO nFit: alpha(j, k) = covar(j, k): NEXT k
         da(j) = beta(j)
      NEXT j
      FOR j = 1 TO nParm: A(j) = atry(j): NEXT j
   ELSE
      aLamda = 10 * aLamda
      chisq = ochisq
   END IF

   mrqmin = 0
Mrqexit:
’ERASE atry, beta, ochisq, j, k, l, m, nErr
ERASE atry, beta

END FUNCTION ’mrqmin

REM $STATIC
SUB Plot (mode, Flnm$, psc)
’Plot -- Subroutine that plots data on screen, sends it to the HP plotter, and
’           then uses the Fit routines to fit interferometer signals to sine
’           waves.
```

```
’
’Recoil measurement constants
CONST k1 = 1117841.917# ’F=3 -> F’=3 wavenumber / (2*pi) in m^-1
CONST k2 = 1117811.254# ’F=4 -> F’=3 wavenumber / (2*pi) in m^-1
CONST keff = k1 + k2 ’Effective wavenumber / (2*pi) in m^-1
CONST hmCs = 3.002370019D-09 ’h/mCs in m^2/s
CONST fRec = 2 * hmCs * keff * keff ’Recoil splitting for 4 pi/2 interferometer

’Miscellaneous constants
CONST nXpts = 640, nYpts = 480
’*60:
CONST nFtPts = 100
CONST fmtA$ = "A:##.###", fmtAsd$ = "_+_-#.###", fmtAexp$ = "E+#"
CONST fmtB$ = "B:##.###"
CONST fmtC$ = "C:###.#", fmtCsd$ = "_+_-##.#%"
CONST fmtk$ = "k:##.###", fmtksd$ = "_+_-#.###"
CONST fmtp$ = "p:##.###", fmtpsd$ = "_+_-#.####rad"
CONST fmtch$ = "chi2:##.###^^^^"
’CONST fmtf$ = "+##.####", fmtfsd$ = "_+_-#.#### Hz" ’, fmtff$ = " (+#.#^^^^)"

’$DYNAMIC
DIM parm(1 TO nParm, 1 TO nSeqMax), psd(1 TO nParm, 1 TO nSeqMax)
DIM Aexp(1 TO nSeqMax), scA(1 TO nSeqMax), Bexp(1 TO nSeqMax), scB(1 TO nSeqMax)
DIM chi(1 TO nSeqMax), nErrFit(1 TO nParm)

      ’Prepare to print summary results
      COLOR 15 ’Bright white print
      ’Check whether file exists.  If so, read in data.
      ON ERROR GOTO 999 ’Error for file not found
      iErr = 0
      FlnmP$ = Flnm$
      OPEN FlnmP$ FOR INPUT AS 1
      IF iErr THEN
         PRINT FlnmP$; " not found. Press any key to continue."
         DO: LOOP WHILE INKEY$ = ""
         GOTO 200
      END IF

      ON ERROR GOTO 99 ’Error routine for reading past end of file

      IF ABS(psc) < .0001 THEN psc = 1

      ’Read in number of steps and number of sequences
      INPUT #1, nSteps, nSeq
      IF iErr GOTO 200

      ’Arrays for frequency and amplitude data
      DIM freq(nSteps, 1 TO nSeq), ampl(nSteps, 1 TO nSeq), cor(nSteps, 1 TO nSeq)
      DIM idisc(nSteps, 1 TO nSeq)
      DIM Bck(1 TO nSeq)
      ’Read in signal data
      FOR i = 0 TO nSteps
         LINE INPUT #1, A$
         IF i = 0 THEN ’Determine formatting of data
            flen% = 1 ’Determine length of frequency data
            DO WHILE MID$(A$, flen%, 1) = " ": flen% = flen% + 1: LOOP
            DO WHILE MID$(A$, flen% + 1, 1) <> " ": flen% = flen% + 1: LOOP
            tlen% = flen% + 1 ’Determine length to end of amplitude data
            DO WHILE MID$(A$, tlen%, 1) = " ": tlen% = tlen% + 1: LOOP
            DO WHILE MID$(A$, tlen% + 1, 1) <> " ": tlen% = tlen% + 1: LOOP
            IF nSeq = 1 THEN
               clen% = LEN(A$) ’Total data length
            ELSE
              clen% = tlen% + 1 ’Determine length to end of freq. correction
              DO WHILE MID$(A$, clen%, 1) = " ": clen% = clen% + 1: LOOP
              DO WHILE MID$(A$, clen% + 1, 1) <> " ": clen% = clen% + 1: LOOP
            END IF
            alen% = tlen% - flen%
            blen% = clen% - tlen%
         END IF
         IF iErr GOTO 200
```

```
         FOR j = 1 TO nSeq
'           iStr = tlen% * (j - 1) + 1 'First string character for Seq. #j
            iStr = clen% * (j - 1) + 1 'First string character for Seq. #j
            freq(i, j) = VAL(MID$(A$, iStr, flen%))
            ampl(i, j) = VAL(MID$(A$, iStr + flen%, alen%))
            freq(i, j) = freq(i, j) + VAL(MID$(A$, iStr + tlen%, blen%))
              'correct frequency for synthesizer phase errors
         NEXT j
      NEXT i

      'Read in nPi and T (for curve fit)
      INPUT #1, nPi, Tint
      IF iErr GOTO 200
      phisc = 1 / (twopi * Tint * (nPi + 1)) 'phase->frequency scale factor

      'Read in parameters needed for PLOT40
      FOR j = 1 TO nSeq
         INPUT #1, mSeq(j), fCAve#(j)
         IF iErr GOTO 200
      NEXT j

      IF mode THEN CLS 'CLS when calling from Menu

      'Read in graphing parameters and numbers of statistics
      INPUT #1, yscP, k0, nStat
      IF iErr GOTO 200
'*60      yscP = yscP / 10 ^ k0

      'Read in statistics
      FOR j = 1 TO nStat
         LINE INPUT #1, A$
         'Read in average background value(s)
         IF LEFT$(A$, 11) = "Ave Backgnd" THEN
           FOR j2 = 1 TO nSeq
             Bck(j2) = VAL(MID$(A$, 1 + 13 * j2, 13))
           NEXT j2
         END IF
         IF iErr GOTO 200
         PRINT A$
      NEXT j

      'Read in lengths of frequency list and parameter list
      INPUT #1, nList, nSumm
      IF iErr GOTO 200

      IF mode THEN 'List frequency list with other summary data
         nSumm = nSumm + nList + 1
      ELSE 'Read in frequency list, but don't print out
         FOR j = 0 TO nList
            LINE INPUT #1, A$
            IF iErr GOTO 200
         NEXT j
      END IF

      'Read in and print out summary data
      FOR j = 1 TO nSumm
         LINE INPUT #1, A$
         IF iErr GOTO 200
         PRINT A$
         IF (CSRLIN > 27) OR j = nSumm THEN
            PRINT : PRINT "Press any key to continue.";
            DO: A$ = INKEY$: LOOP WHILE A$ = ""
            CLS : LOCATE 1, 1
         END IF
      NEXT j
      CLOSE 1
      ON ERROR GOTO 0 'Standard error routine

'*61b
  'find maximum value of data
  amplmx = ampl(0, 1)
  FOR j = 1 TO nSeq
    FOR i = 0 TO nSteps
      IF (ampl(i, j) > amplmx) THEN amplmx = ampl(i, j)
    NEXT i
  NEXT j
  dum = LOG(ABS(amplmx)) / LOG(10#)     'log10()
  yscP = INT(dum)                 'nearest decade = X
  dum2 = 10 ^ (dum - yscP)        'factor within the decade
  yscP = 10 ^ yscP                'scaling factor (graph maximum) = 10^X = 1eX
  IF (dum2 > 5.1) THEN
    yscP = yscP * 10              'scaling factor 10eX
  ELSEIF (dum2 > 2.1) THEN
    yscP = yscP * 5               'scaling factor 5eX
  ELSEIF (dum2 > 1.1) THEN
    yscP = yscP * 2               'scaling factor 2eX
  END IF
'*61e

100   CLS : GOSUB border'Draw plot borders and center lines
      'Plot points
      iPts = -1: iMode = 1: j0 = 1: GOSUB graph

      DO
        LOCATE 14, 1
        PRINT "Press any key to continue, 's' to change scale factor";
'*55        IF ((mSeq(1) <> 0) AND (NOT iFit)) THEN
          PRINT ", 'f' to fit        ";
'*55        ELSE
'*55          PRINT "                    ";
'*55        END IF
        DO: q$ = UCASE$(INKEY$): LOOP WHILE q$ = ""
        IF q$ = "S" THEN
          LOCATE 14, 1: PRINT STRING$(78, 32); : LOCATE 14, 1
          PRINT "Scale factor for plot is"; STR$(yscP);
          INPUT ".  New scale factor = (# or RET)"; A$: A = VAL(A$)
          IF A$ <> "" AND A > 0 THEN
            iMode = 0: GOSUB graph 'Erase plot with old scale factor
            yscP = A: iMode = 1: GOSUB graph 'Redraw plot with new scale factor
          END IF
'*55        ELSEIF q$ = "F" AND (mSeq(1) <> 0) AND (NOT iFit) THEN
'*60b
        ELSEIF q$ = "F" AND (NOT iFit) THEN
          IF (mSeq(1) <> 0) THEN
'*60e
            CALL Fit2(Tint, nSteps + 1, nSeq, freq(), ampl(), idisc(), parm(), psd(), chi(), nErrFit
())
            'Find scaling for printing signal amplitude
            FOR j = 1 TO nSeq
                IF parm(1, j) <> 0 THEN
                  Aexp(j) = INT(LOG(ABS(parm(1, j))) / LOG(10))
                ELSE
                  Aexp(j) = 0
                END IF
                scA(j) = 10 ^ -Aexp(j)
                IF parm(2, j) > parm(1, j) THEN
                  IF parm(2, j) <> 0 THEN
                     Bexp(j) = INT(LOG(parm(2, j)) / LOG(10))
                  ELSE
                     Bexp(j) = 0
                  END IF
                  scB(j) = 10 ^ -Bexp(j)
                END IF
            NEXT j
        LOCATE 2, 1
        FOR m1 = 1 TO 3 STEP 2 'Look for 1->2 and 3->4 pairs
         IF (m1 = 3) THEN LOCATE 3, 1
         iHead = -1 'Print heading before next pair
         FOR j = 1 TO nSeq
          IF mSeq(j) = m1 AND nErrFit(j) = 0 THEN
            FOR k = 1 TO nSeq 'Look for corresponding interferometer
              IF mSeq(k) = m1 + 1 AND nErrFit(k) = 0 THEN
                IF iHead THEN
                  PRINT m1; "->"; m1 + 1; ":";
                  iHead = 0
                ELSE
                  PRINT ", ";
                END IF
                dphi = parm(4, j) - parm(4, k)
                dpsd = SQR(psd(4, j) ^ 2 + psd(4, k) ^ 2)
                nErr = mod2pi(dphi)
                IF nErr THEN PRINT "*"; LTRIM$(STR$(nErr)); "*";
                IF fCAve#(j) > fCAve#(k) THEN dphi = -dphi
                dfRec = phisc * dphi
                dfsd = phisc * dpsd
                PRINT USING "+##.####"; dfRec;
                PRINT USING "_+_-#.#### Hz  = "; dfsd;
                PRINT USING "+###.##"; dphi * 1000;
                PRINT USING "(###.##) mrad  = "; dpsd * 1000;
                dfRec = dfRec / fRec * 1E+09
                dfsd = dfsd / fRec * 1E+09
                IF ((ABS(dfRec) < 10000) AND (dfsd < 10000)) THEN
                  PRINT USING "+####.#"; dfRec;
                  PRINT USING "(####.#) ppb"; dfsd;
                ELSE
                  PRINT USING "+###.##"; dfRec / 1000;
                  PRINT USING "(###.##) ppm"; dfsd / 1000;
                END IF
              END IF
            NEXT k
          END IF
         NEXT j
'         PRINT
        NEXT m1
        iFit = -1
        GOSUB border 'Redraw border
        iPts = -1: GOSUB graph 'Now plot curves, and data
'*55        iPts = -1: 'From now on, plot both
'*60b
          ELSE
            CALL FitGhead(nSteps + 1, freq(), ampl(), idisc(), parm(), psd(), chi(), nErrFit())
            iFit = -1
            GOSUB border                'Redraw border
            iPts = -1: GOSUB graph      'Now plot curves, and data
          END IF
'*60e
        END IF
      LOOP WHILE q$ = "S" OR q$ = "F"

200   CLS
      ON ERROR GOTO 0

ERASE parm, psd, Aexp, scA, Bexp, scB, chi, nErrFit
ERASE freq, ampl, cor, Bck

      PRINT "Leaving Plot"

      EXIT SUB

border: 'Draw plot border and center line
      IF mSeq(1) = 0 THEN 'Single plot
         LINE (0, 384)-(639, 384), 15: LINE (320, 0)-(320, 479), 15
         SCX = (nXpts - 1) / nSteps: SCY = nYpts - 1
'*60:
         SCXf = (nXpts - 1) / (nFtPts - 1)
      ELSE 'Four plots
         LINE (320, 0)-(320, 479), 9: LINE (0, 240)-(639, 240), 9
         LINE (0, 192)-(639, 192), 15: LINE (0, 432)-(639, 432), 15
         LINE (160, 0)-(160, 479), 15: LINE (480, 0)-(480, 479), 15
         SCX = (nXpts - 1) / (2 * nSteps): SCY = (nYpts - 1) / 2
'*60:
         SCXf = (nXpts / 2 - 1) / (nFtPts - 1)
      END IF
      LINE (0, 0)-(639, 479), 9, B 'Border box
      'Display background value
      IF mSeq(1) = 0 THEN
        LOCATE 29, 71: PRINT USING "##.#^^^^V"; Bck(1);
      ELSE
        FOR j = 1 TO nSeq
          m = mSeq(j)
          iyC = 1 + 14 * INT((m + 1) / 2): ixC = 71 - 40 * (m MOD 2)
          LOCATE iyC, ixC: PRINT USING "##.#^^^^V"; Bck(j);
        NEXT j
      END IF
      RETURN

graph: 'Plot the data and fit results on the screen
      COLOR 14
      LOCATE 30, CINT(41 - LEN(Flnm$) / 2): PRINT Flnm$;
      COLOR 15
      LOCATE 1, 38: PRINT USING "#^^^^V"; yscP; 'display main plot scale
      IF iMode = 0 THEN iColP = 0 'Erase
      IF iMode = 1 THEN iColP = 11
      FOR jj = 1 TO nSeq
         'Plot in reverse order with j0 last
         j = ((j0 + 2 * nSeq - 1 - jj) MOD nSeq) + 1
         IF iMode = -1 THEN 'Selecting interferometers
            IF j = j0 THEN
               iColP = 15 'Bright white for current selection
            ELSE
               iColP = 11
            END IF
         END IF
         m = mSeq(j): IF m = 0 THEN m = 1 'm -> location of plot
         mSw = INT((m + 1) / 2) 'Row for plot

         IF iFit THEN 'Print fit parameters
'*60:
       IF (mSeq(1) <> 0) THEN           'sine wave fit
          IF iMode = -2 THEN
            COLOR 0
          ELSE
            IF iMode = -1 AND j = j0 THEN
               COLOR 15
            ELSE
               COLOR 7
            END IF
          END IF
          ix = 42 - 40 * (m MOD 2)
          iy = 14 * mSw - 10
          LOCATE iy, ix
          IF nErrFit(j) = 0 THEN 'Valid fit -> list parameter values
            PRINT USING fmtA$; parm(1, j) * scA(j);
            PRINT USING fmtAsd$; psd(1, j) * scA(j);
            PRINT USING fmtAexp$; Aexp(j)
            LOCATE iy + 1, ix
            IF parm(2, j) > parm(1, j) THEN 'Print as amplitude
               PRINT USING fmtB$; parm(2, j) * scB(j);
               PRINT USING fmtAsd$; psd(2, j) * scB(j);
               PRINT USING fmtAexp$; Bexp(j)
            ELSE 'Print as contrast
               Cntrst = 100 * parm(2, j) / parm(1, j)
               CntrstSD = Cntrst * SQR((psd(1, j) / parm(1, j)) ^ 2 + (psd(2, j) / parm(2, j)) ^ 2)
               PRINT USING fmtC$; Cntrst;
               PRINT USING fmtCsd$; CntrstSD
            END IF
            'LOCATE iy + 2, ix: PRINT USING fmtk$; parm(3, j);
            'PRINT USING fmtksd$; psd(3, j)
            LOCATE iy + 2, ix: PRINT USING fmtp$; parm(4, j);
            PRINT USING fmtpsd$; psd(4, j)
            LOCATE iy + 3, ix: PRINT USING fmtch$; chi(j)
          ELSE 'List error number
            PRINT "Fit: Error #:"; nErrFit(j)
          END IF
```

```
'*60b
      ELSE                              'gaussian fit
        COLOR 7
        ix = 2
        iy = 2
        LOCATE iy, ix
        IF nErrFit(j) = 0 THEN 'Valid fit -> list parameter values
          PRINT "Fit function:"
          LOCATE iy + 1, ix + 2
          PRINT "A exp{-0.5*[(f-f0)/B]^2} + C"
          LOCATE iy + 2, ix + 4
          CALL PUNice(parm(1, j), psd(1, j), " A", " V")
          LOCATE iy + 3, ix + 4
          CALL PUNice(parm(2, j), psd(2, j), "f0", " Hz")
          LOCATE iy + 4, ix + 4
          CALL PUNice(parm(3, j), psd(3, j), " B", " Hz")
          LOCATE iy + 5, ix + 4
          CALL PUNice(parm(4, j), psd(4, j), " C", " V")
        ELSE 'List error number
          PRINT "Fit: Error #:"; nErrFit(j)
        END IF
      END IF
'*60e
        END IF
     IF iMode <> -2 THEN 'Plot data and/or curve
         ix0 = ((m - 1) AND 1) * nSteps
         y0 = SCY * (mSw - .2)
         IF iPts THEN 'Plot data points
'*60:
          fmn = freq(0, j): fmx = freq(0, j)
          FOR i = 0 TO nSteps
'*60b
           IF (freq(i, j) < fmn) THEN fmn = freq(i, j)
           IF (freq(i, j) > fmx) THEN fmx = freq(i, j)
'*60e
           dum = ampl(i, j) / yscP
           IF dum > -.25 AND dum < 1 THEN
              ix = CINT(SCX * (ix0 + i))
              iy = CINT(y0 - .8 * SCY * dum)
              PSET (ix, iy), iColP
              IF idisc(i, j) = 1 THEN CIRCLE (ix, iy), 4, iColP: PSET (ix, iy), 15
           END IF
          NEXT i
         END IF

         IF iFit AND nErrFit(j) = 0 THEN 'Plot curve
'*60b
           IF ((m = 2) OR (m = 4)) THEN
             ix0 = nXpts / 2
           ELSE
             ix0 = 0
           END IF
           fstp = (fmx - fmn) / (nFtPts - 1)
'*60e
           iLine = 0 'PSET for first point
'*55           FOR i = 0 TO nSteps
'*55:60b             dum = (parm(1, j) + parm(2, j) * COS(twopi * Tint * parm(3, j) * freq(i, j) + p
arm(4, j))) / yscP
           FOR i = 0 TO (nFtPts - 1)
            ff = fmn + fstp * i
            IF (mSeq(1) <> 0) THEN          'sine wave fit
              dum = (parm(1, j) + parm(2, j) * COS(twopi * Tint * parm(3, j) * ff + parm(4, j))) /
yscP
            ELSE
              dum = (parm(1, j) * EXP(-.5 * ((ff - parm(2, j)) / parm(3, j)) ^ 2) + parm(4, j)) / y
scP
            END IF
'*60e
            IF dum > -.25 AND dum < 1 THEN
'*55:60               ix = CINT(SCX * (ix0 + i))
              ix = CINT(ix0 + i * SCXf)
              iy = CINT(y0 - .8 * SCY * dum)
              IF iLine THEN
                 LINE -(ix, iy), iColP
              ELSE
                 PSET (ix, iy), iColP
              END IF
              iLine = -1 'Draw line to next point
            ELSE
              iLine = 0  'PSET next point
            END IF
           NEXT i
         END IF
      END IF
      NEXT jj

      COLOR 15
      RETURN

END SUB 'Plot

REM $STATIC
SUB PUNice (x, xsd, xnm$, xunit$)
'Print variable xnm$ with value x and error bar xsd in a nice form
'$DYNAMIC
DIM xexp, xsc

  IF x <> 0 THEN
    xexp = INT(LOG(ABS(x)) / LOG(10))
  ELSE
    xexp = 0
  END IF
  xsc = 10 ^ -xexp

  PRINT xnm$; ":";
  PRINT USING "##.###"; x * xsc;
  PRINT USING "_+_-#.###"; xsd * xsc;
  PRINT USING "E+#"; xexp;
  PRINT xunit$;

END SUB

REM $STATIC
' Put parameters in "standard" form
SUB standardform (j, p(), psd())

DIM p1

  IF p(3, j) < 0 THEN
      p(3, j) = -p(3, j): p(2, j) = -p(2, j): p(4, j) = -p(4, j)
  END IF
  IF p(2, j) < 0 THEN p(2, j) = -p(2, j): p(4, j) = p(4, j) + pi
  IF mod2pi(p(4, j)) AND p1% >= 10 THEN
   PRINT "WARNING: parameter shifted by a multiple of 2pi"
  END IF

'ERASE p1

END SUB
```

C.2 Fit.C

The Fit.C code compiled in *Borland* Turbo C++ 3.0 and running under DOS takes the output data files from the AltInt.BAS program described in Section C.1 and fits the interferometer fringes using the non-linear least-squares fit algorithm based on the Levenberg-Marquardt method of root finding [50]. It can fit two conjugate interferometer fringes simultaneously and also display the results graphically. All of our interferometer data was fit using this code.

```
#define Program __FILE__
/*
  FitXX.C                                Joel Hensley

Revision History
1.1 10/9/96 Sign of calculated recoil corrected.
            Precision of check for agreement between known recoil shift
              and difference of center frequencies changed from 0.01 to
              0.0001
            When autofitting, 2 chi^2 values are produced.  One from the
              1st pass where AutoVary parameter(s) are not varied, and one
              from the 2nd pass where AutoVary parameter(s) are allowed to
              vary.  Before comparing these chi^2 values they must be
              properly scaled by the number of data points minus the number
              of free parameters.
            Remove display of signal-to-noise (S/N) from summary and
              SigmaPlot files.
            Do not perform modulo(2Pi) on error in phase parameter
1.2 12/4/96 The fit routines will crash if any of the amplitude values are
              negative.  This happens when the background values are wrong.
              Fix this problem by checking for negative values and shifting
              all values up until they are all non-negative.
             Fix display colors.
             Improve output format in fit subsection.
             Make separate subroutine that calls the fit routines, so that
              multiple-pass fitting does not repeat code.
1.3 12/6/96  Use correct method for dropping out 3-sigma points: run fit
              once to get an initial set of fit parameters, use these initial
              parameters to calculate residuals and a standard deviation of
              the residuals, throw out any data points that are more than 3
              (or 5) standard deviations away from this initial fit curve,
              refit using this reduced set of data points.
             Improve "SigmaPlot" output file format
             Convert large static 2D arrays to dynamic arrays using malloc()
             Read in time data was taken from input data file
1.4 1/23/97  Increased maximum data file name length from 30 to 256
             Added column to summary and SigmaPlot output files: the ratio
               between the fit amplitudes (parameter A), A1/A2, A4/A3, where
               Ai is the amplitude of the interferometer #i.
1.5 2/10/97  Added ability to correct the final recoil shift for RF phase
               errors.  A third order polynomial is used to calculate the
               phase shift from RF components at the frequencies of the
               third and fourth Pi/2 pulses for each conjugate
               interferometer.  These phase shifts are then added with the
               correct signs into the final recoil shift.  The polynomial
               coefficients come from a fit to the experimentally measured
               phase shifts of the relevant RF components.
1.6 11/ 5/97   To correct for the phase error of the new DDS synthesizer
             (ADS-431), the ALTINT program calculates a correction to each
             frequency value of an interferometer scan.  This correction is
             stored as a new column so that data are of the form FREQ, AMP,
             COR for each sequence in the data file.  This version of FIT
             can read this new column and immediately correct the frequency
             values.
               Disable the previous version's "corrected" output.
1.7  1/10/98   Add the preselection offset frequency and Raman sequence
             start time to the summary and SigmaPlot file output.  Increase
             the precision of T and Tram in the summary and sigma plot files.
1.8  3/11/98   Since an absolute measurement of the recoil depends of course
             on the "constant" accepted value that is subracted off of all
             of the frequency differences, this value should be displayed in
             all of the output files.
1.9  4/ 2/98   The data taking program now acquires and stores the table tilt
             from the tilt sensor into the output files.  Read these values
             in and store them to the summary and SigmaPlot files.
               Disable all displays of the "corrected" frequency value (see
             1.6).  Stop displaying the amplitude ratio in the summary file.
2.0  7/ 8/98   The one-sigma uncertainty value for the amplitude parameter
             was not being returned to the original scale.  Now, multiply it
             by "yscale" before returning from fit section.
2.1  7/27/98   Include a display of the phase parameter from the fit of each
             interferometer.  Before, I just displayed the uncertainties and
             then the difference (in Hz) of the phase parameters.
               Replace all expressions of the form A*A with sqr(A*A) which is
             #defined as a macro at the begining.
2.2 11/ 3/99   Correct definition of ysig and ressd
2.3  2/ 4/01   Set fit booleans only at beginning of program not after
             loading each file.  This way, if the fit booleans are changed
             for one file they will stay that way for the rest of the files,
             unless of course they are changed again.
2.4  5/20/01   Temporarily disable the section which looks for the tilt
             sensor data so that we can fit old data.
               Even if the fixed value for the recoil shift is the same in
             this program and the program that takes the data, the difference
             in center frequencies Dcf(acutal) will not exactly agree with the
             predicted values Dcf(desired) because of the discrete resolution
             of the DDS synthesizer.  This program corrects for this by
             modifying the UP/DOWN difference Df/(N+1) by the difference
             between the actual and desired center frequencies:
                 [Dcf(actual) - Dcf(desired)]/(N+1).
             Unfortunately, all previous versions of this code performed this
             correction incorrectly!  They modified the UP/DOWN difference Df
             by: [Dcf(actual) - Dcf(desired)].  This version corrects this
             error.
2.5  7/ 8/01    References to interferometer # (iref) do not work well when
             one interferometer is repeat in a sequence (ex. 1,1).  Change
             function of number keys '1','2','3','4' to refer to sequence
             number instead of interferometer number.  Correct autocommand
             string generator accordingly.  Change the way the '.sp' file is
             generated.  Before, the '.sp' displayed the results from each
             interferometer and printed blanks if that int was not present.
             This does not work for repeat ints in one sequence.  Now, sort
             list of int references (sort not currently implemented) in
             increasing order and display save them in that order in the
             '.sp' file.
*/

#define docorr  (2)     // use correction? 0 = no, 1 = old method, 2 = new

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<process.h>
#include<time.h>
#include"nrutil.h"

enum boolean {FALSE,TRUE};
#define sqr(A)  (A*A)
enum varytype {NO,YES,AUTO};
enum dots {POINT,DOT1,DOT2,DOT3,CROSS,TICKX1,TICKY1,TICKX2,TICKY2,LINE};

#define Pi      (3.141592654)
#define BGIDIR  "e:\\PRG\\TC\\BGI"

#if (docorr)
  #define       SUMFN   "FIT.SUM"       // summary file name
  #define       SPFN    "FIT.SP"        // sigma-plot file name
#else
  #define       SUMFN   "UCF.SUM"       // summary file name
  #define       SPFN    "UCF.SP"        // sigma-plot file name
#endif
#define INFN    "FIT.IN"        /* input file name */
#define VER     1               /* version number */
#define NFITMX  1000            /* maximum # intervals on fit function */

/* calculated value of recoil shift (Hz): */
//#define RECSHFT       (30012.55878090934)
//modified 3/5/98 to include better value for mCs:
```

```
#define RECSHFT         (30012.55775)

FILE
    *in_fp,             /* input file pointer */
    *dat_fp,            /* input data file pointer */
    *out_fp,            /* output file pointer */
    *sum_fp,            /* summary file pointer */
    *sp_fp;             /* sigma-plot file pointer */
int
/* graphics screen constants: */
  ScrnX,ScrnY,          /* graphics screen maximum dimensions */
  NColor,               /* number of graphics colors */
/* plot window paramaters and variables */
  nxdiv = 10,           /* number of divisions along x-axis */
  nxsubdiv = 5,         /* number of subdivision along x-axis */
  nydiv = 8,            /* number of divisions along y-axis */
  nysubdiv = 5,         /* number of subdivisions alonx y-axis */
  winx1,winy1,winx2,winy2,      /* graphics coordinates for plot window */
  fitwx1,fitwx2,fitwy1,fitwy2,  /* graphics coordinates for fit parameter window */
  isauto = FALSE,               /* automatically enter commands? */
  nautocommand = 0,             /* number of auto-commands in string */
  **badpt,                      /* array of indeces to thrown out points */
/* fit menu table dimensions */
  ncol[3] = {2,1,2},
  nrow[3] = {4,3,3};

float
  **of,                         /* offset frequency from input file */
  **amp,                        /* amplitude from input file */
  *xdata,*ydata,*sdata,         /* raw data to fit */
  *res,                         /* residuals from fit */
  **parm,                       /* parameters in fit */
  **psd,                        /* standard deviations for fit parameters */
  leftx,stepx,intx,
  boty,stepy,inty,              /* real coordinates for plot window */
  ymin[4],ymax[4],yoff1,yoff2,  /* minimum value of data */
  xzero[4],                     /* average zero crossing point (rad) */
  ymean[4], ysig[4];            /* mean and standard deviation of data */

char
  ffstr[25] = "A {1 + B sin[C(x + D)]}",
  autocommand[100] = "";        /* automatic command string */



int fgetline(FILE *f_fp, char *line, int *l);
void myfunc(int ids, float x,float *parm,float *y,float *dydp,int nparm);
void plot(float *x, float *y, int n, int *bp, int nb, int color, int dot);
void setextrema(float *x, float *y, int n);
void drawgrid(char *xunits, char *yunits);
void changescale(float *step, float *interval, int dir);
void dataerr(int l);
void tablemove(int it, int dr, int dc, int *r, int *c);
int mod2Pi(float *phi,float cent);
int mod2Pid(double *phi,double cent);
char getautoc(int *i);
void beep(int i);
void standardform(float p[], float psd[]);
void bye(int err, int ns);
void FreeDynam(int ns);
float RunFit(int dodisp,
             int n, float x[], float y[], float s[],
             int nparm, float p[], int pvary[],
             int dosimfit, int idsref[], float yscale,
             float **covar, float **alpha);
unsigned long TimeToSeconds(char *s);
float RFphase(double f);
void DateTimeStr(time_t t, char *ds, char *ts) {
/* Convert the time "t" into mm/dd//yy hh:mm:ss format. */
  struct tm* lt;
```

```
  lt = localtime(&t);
  sprintf(ds,"%02i/%02i/%02i",lt->tm_mon+1,lt->tm_mday,lt->tm_year);
  sprintf(ts,"%02i:%02i:%02i",lt->tm_hour,lt->tm_min,lt->tm_sec);
}
/* external routines: */
extern int strfind(char *s1, char *s2);
int isnum(char c)       // is "c" part of a number?
{
  return ((c=='+') || (c=='-') || (c=='.') || isdigit(c));
}


int main(int narg, char **argstr)
{
  char
    datfn[30],          /* data file name */
    outfn[30],          /* output file name */
    ds[9],ts[9],        // date and time
    xunits[10] = "Hz",
    yunits[10] = "(arb)",
    varystr[3][5] = {" No "," Yes","Auto"},
    parmname[8][2] = {"","A","B","C","D","A","B","D"};

  int
    graphdriver = DETECT, graphmode,
    version,            /* version number for input file format */
    ns,nstep,           /* number of steps in each interferometer sequence */
    ni,nseq,            /* number of interferometer sequences */
    nis[4],             // indeces to sorted list of interferometers
    iref[4],            /* interometer sequence reference code */
    dohead = TRUE,      // should I print a header for the _sp file?
    conj[4] = {1,0,3,2}, /* ref. numbers for the conjugate interferometers */
    hasint[4]={-1,-1,-1,-1},hi1,hi2,hi3,hi4,
                        /* which interferometer sequences are present? */
    npi,                /* number of Pi pulses */
    inti = 0, intj,     /* current and conjugate interferometer number (0-3) */
    nxpz, nxnz,         /* # of pos., neg. going zero-crossings */
    istext = TRUE,      /* is current screen textual (or graphical)? */
    donegraph = FALSE,  /* already plotted data? */
    donefit[4] = {FALSE,FALSE,FALSE,FALSE},
                        /* already fit the data for each interferometer? */
//*24    dosimfit,              /* do simultaneous fit of two data sets? */
//*25:
    dosimfit, dsf0 = TRUE,      // do simultaneous fit of two data sets?
    dispparm = FALSE,   /* show parameter values after each fit iteration */
    idsref[3],          /* index of 1st data point in each data set */
    showfit = FALSE,    /* display the fit? */
    itable,             /* currently active fit options table:
                             0: primary interferometer fit options
                             1: general options
                             2: conjugate interferometer fit options */
    nparm,              /* number of parameters in model */
    pvary[8],pvinit[8], /* how this parameter is varied in fit:
                              0 = NOT varied
                              1 = varied
                              2 = not varied, then varied */
    n2pass,             /* number of parameters varied in 2 passes */
    ndata,              /* # of data points within 3-sigma to use in fit */
    nignore,            /* # of points thrown out for a given inter. */
    nbad[4] = {-1,-1,-1,-1},  /* # of thrown out points for each inter. */
    ndpfit[4],          /* number of points used in each fit */
    nfpfit[4],          /* number of free parameters used in each fit */
    row,col,            /* row and column in initial fit-parameter table */
    locx[3][2] = {{16,27},{48,48},{41,52}},        /* positions of columns */
    locy[3][4] = {{3,4,5,6},{8,9,10,9},{3,4,6,6}}, /* positions of rows */
    npts,               /* number of intervals used when plotting fit */
    color,              /* color to draw points */
    iac = 1;            /* auto-command index */

  float
    TT,                 /* time between Pi pulses */
```

```
    cor,                /* correction to a frequency value (Hz) */
    xpz, xnz,           /* positive and negative going zero-crossings */
    tltx, tlty,         // x and y-axis tilt values (V)
    yscale,             /* y scaling factor */
    parminit[8],        /* initial values of fit parameters */
    fitparm[8],         /* parameter array used by fit */
    oldparm[8],pval,    /* scratch space to save parameter values */
    oldvar[8],          /* scratch space to save covariances */
    pchisq[4],          /* chi-squared value from fit */
    dydp[8],            /* derivatives with respect to the paramaters */
    **covar,            /* covariance matrix */
    **alpha,            /* curvature matrix */
    chisq,newchisq,
    resmean,ressd,      /* mean and standard deviaiton of residuals */
    resfact = 3.0,      /* # of sigma away before a point is thrown out */
    fstepx,
    xfit[NFITMX],
    yfit[NFITMX],
    ar12,ar43,          /* fit amplitude ratios */
    rf1,rf2;            /* RF phase shifts for each conjugate int (cycles) */

  unsigned long
    ltm = 0;            /* time data was taken (seconds from 00:00:00) */

  double
    p2f3[4], p2f4[4],   // frequencies of 3rd, 4th Pi/2 pulses (Hz)
    fpre,               // preselection offset frequency (Hz)
    tram,               // Raman sequence start time (ms)
    f0,                 // integer multiple of the "fixed" recoil freq. RECSHFT
    df,ddf,ddfs,        // recoil shift and recoil shift error error-bar
    cddf,               // recoil shift correct for RF phase error
    cf[4];              // center frequency

  time_t tim;
  char c,s1[80],*c_p;
  int i,j,k,l,bool1,bool2;
  float f1,f2,f3;
  double d1;
//temporary patch
//  const double tempcor[5]={0.,-0.0014,-0.2442, 0.0154,-0.0001};       // 7/20
//  const double tempcor[5]={0., 0.0006,-0.2443, 0.0154,-0.0001};       // 7/21
//  const double tempcor[5]={0., 0.0004,-0.2444, 0.0155, 0.0000};       // 7/22
//temporary patch

  clrscr();
  textcolor(LIGHTGRAY);

/* open summary file */
  if ((sum_fp=fopen(SUMFN,"wt")) == NULL) {
    cprintf("fit: ERROR opening summary ouput file '%s'.\r\n",SUMFN);
    bye(1,0);
  }
  tim = time(NULL);
  DateTimeStr(tim,ds,ts);
  fprintf(sum_fp,"Summary file created by %s on %s at %s\n",Program,ds,ts);
  fprintf(sum_fp,"f0 = (N+1) * %.11lf Hz.\n",(double)RECSHFT);
  fprintf(sum_fp,"DataFile   T(ms)   Npi Fpre(Hz)  Tram(ms)\n");
  fprintf(sum_fp,"    Contrast(%)     Phase(rad)       k'     chi^2   N :Nparm\n");
  fflush(sum_fp);
/* open sigma-plot file */
  if ((sp_fp=fopen(SPFN,"wt")) == NULL) {
    cprintf("fit: ERROR opening sigma-plot ouput file '%s'.\r\n",SUMFN);
    bye(1,0);
  }
  fprintf(sp_fp,"FileName,T(ms),N_Pi,Fpre(Hz),Tram(ms),");
/* read input file */
  if ((in_fp=fopen(INFN,"rt")) == NULL) {
    cprintf("fit: ERROR opening input file '%s'.\r\n",INFN);
    bye(1,0);
  }
  cprintf("Reading program input file.\r\n");
```

```
  fscanf(in_fp,"%i\n",&version);
  if (version != VER) {
    cprintf("fit: input file '%s' has wrong version number.\r\n",INFN);
    bye(1,0);
  }
  /* default values for fit code: */
  npts = 500;
  nparm = 4;
  pvinit[1] = YES;
  pvinit[2] = YES;
  pvinit[3] = NO;
  pvinit[4] = YES;
  pvinit[5] = YES;
  pvinit[6] = YES;
  pvinit[7] = YES;
  do {
    l = 30;
    if (fgetline(in_fp,datfn,&l)) {
      cprintf("fit: ERROR reading input file.\r\n");
      bye(1,0);
    }
    if (l > 0) {
      clrscr();
/* reset initial values */
      hasint[0] = hasint[1] = hasint[2] = hasint[3] = -1;
      inti = 0;
      istext = TRUE;
      donegraph = FALSE;
      donefit[0] = donefit[1] = donefit[2] = donefit[3] = FALSE;
/* read input data file */
      if ((dat_fp=fopen(datfn,"rt")) == NULL) {
        cprintf("fit: ERROR opening input data file, '%s'.\r\n",datfn);
        bye(1,0);
      }
      cprintf("Reading interferometer data from file '%s'.\r\n",datfn);
      fscanf(dat_fp,"%i\n",&nstep);
      if (nstep < 1) dataerr(1);
      fscanf(dat_fp,"%i\n",&nseq);
      if ((nseq < 1) || (nseq > 4)) dataerr(2);
      /* now that we know how many interferometers and data points are in
         this data file, allocate the required storage space */
      bool1 = FALSE;
      j = 2*(nstep+1);
      if ((xdata=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
      xdata--;          /* starting index = 1 */
      if ((ydata=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
      ydata--;          /* starting index = 1 */
      if ((sdata=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
      sdata--;          /* starting index = 1 */
      if ((res=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
      res--;            /* starting index = 1 */
      if ((of=(float **)malloc(nseq*sizeof(float *)))==NULL) bool1 = TRUE;
      if ((amp=(float **)malloc(nseq*sizeof(float *)))==NULL) bool1 = TRUE;
      if ((badpt=(int **)malloc(nseq*sizeof(int *)))==NULL) bool1 = TRUE;
      j = nstep+1;
      if ((parm=(float **)malloc(nseq*sizeof(float *)))==NULL) bool1 = TRUE;
      if ((psd=(float **)malloc(nseq*sizeof(float *)))==NULL) bool1 = TRUE;
      for(i=0; i<nseq; i++) {
        if ((of[i]=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
        if ((amp[i]=(float *)malloc(j*sizeof(float)))==NULL) bool1 = TRUE;
        if ((badpt[i]=(int *)malloc(j*sizeof(int)))==NULL) bool1 = TRUE;
        if ((parm[i]=(float *)malloc(7*sizeof(float)))==NULL) bool1 = TRUE;
        parm[i]--;              /* starting index = 1 */
        if ((psd[i]=(float *)malloc(7*sizeof(float)))==NULL) bool1 = TRUE;
        psd[i]--;               /* starting index = 1 */
      }
      if (bool1) {
        cprintf("fit: ERROR allocating dynamic memory.\r\n");
        bye(1,0);
      }
      /* continue loading in the data */
```

```
      for(ns=0; ns<=nstep; ns++) {
        for(ni=0; ni<nseq; ni++) {
          if (fscanf(dat_fp,"%f%E%f",&(of[ni][ns]),&(amp[ni][ns]),&cor) != 3)
            dataerr(ns+3);
#if (docorr==2)
          of[ni][ns] += cor;
#endif
        }
        fscanf(dat_fp,"\n");
      }
      fscanf(dat_fp,"%i\n",&npi);
/* basic uses #.########D-## format to express double precision reals.
C does not recoqnize the 'D', so if there is an alphanumeric character
in the expression, parse the mantissa and exponents separately." */
      // load T
      l = 30;
      if (fgetline(dat_fp,s1,&l)) dataerr(nstep+5);
      for(i=0; i<l && !isalpha(s1[i]); i++);
      sscanf(s1,"%f",&TT);
      if (i < l) {
        sscanf(s1+i+1,"%i",&j);
        TT = TT * pow(10.0,(float)j);
      }
      // load interferometer reference number and center frequency
      for(ni=0; ni<nseq; ni++) {
        if (fscanf(dat_fp,"%i\n%lf\n",&iref[ni],&cf[ni]) != 2)
          dataerr(nstep+6+ni);
        if ((iref[ni] < 1) || (iref[ni] > 4)) dataerr(nstep+4+ni);
        hasint[iref[ni]-1] = ni;
//temporary patch
//      for(i=0; i<=nstep; i++)
//        of[ni][i] += tempcor[iref[ni]];
//temporary patch
      }
      tltx = 0.;
      tlty = 0.;
      do {      // look for the phrase "X-tilt"
        l = 80;
        fgetline(dat_fp,s1,&l);
      } while (strfind(s1,"X-tilt") == -1);
      for(i=7; !isnum(s1[i]); i++);
      sscanf(s1+i,"%f",&tltx);          // read x-axis tilt
      fgetline(dat_fp,s1,&l);
      for(i=7; !isnum(s1[i]); i++);
      sscanf(s1+i,"%f",&tlty);          // read y-axis tilt
      /* load in the frequencies of the 3rd and 4th Pi/2 pulses */
      do {      /* look for the first occurence of "pi/2 pair:" */
        l = 80;
        fgetline(dat_fp,s1,&l);
      } while (strfind(s1,"pi/2 pair:") == -1);
      do {      /* look for the second occurence of "pi/2 pair:" */
        l = 80;
        fgetline(dat_fp,s1,&l);
      } while (strfind(s1,"pi/2 pair:") == -1);
      for(i=10; isspace(s1[i]); i++);
      for(ni=0; ni<nseq; ni++) {                /* 3rd pi/2 */
        for(; !isspace(s1[i]); i++);
        for(; isspace(s1[i]); i++);
        sscanf(s1+i,"%lf",&p2f3[ni]);
      }
      l = 80;
      fgetline(dat_fp,s1,&l);
      for(i=10; isspace(s1[i]); i++);
      for(ni=0; ni<nseq; ni++) {                /* 4th pi/2 */
        for(; !isspace(s1[i]); i++);
        for(; isspace(s1[i]); i++);
        sscanf(s1+i,"%lf",&p2f4[ni]);
      }
      /* load in time the data was taken: */
      do {      // look for the phrase "Data collected by"
        l = 80;
```

```
        fgetline(dat_fp,s1,&l);
      } while (strfind(s1,"Data collected by") == -1);
      ltm = TimeToSeconds(strchr(s1,':')-2);
      /* load in preselection offset frequency and Raman start time: */
      do {      // look for the phrase "Preselection offset ="
        l = 80;
        fgetline(dat_fp,s1,&l);
        j = strfind(s1,"Preselection offset =");
      } while (j == -1);
      sscanf(s1+j+21,"%lf",&fpre);
      j = strfind(s1,"Raman trigger time =");
      sscanf(s1+j+20,"%lf",&tram);
      fclose(dat_fp);

      cprintf("Input data file: %s\r\n",datfn);
      cprintf("Number of points: %i\r\n",nstep+1);
      cprintf("Number of interferometers: %i\r\n",nseq);

/*
  for(ns=0; ns<=nstep; ns++) {
    for(ni=0; ni<nseq; ni++)
      printf("%13.3f%13.5E",of[ni][ns],amp[ni][ns]);
    printf("\n");
  }
*/
      cprintf("Number of Pi pulses: %i\r\n",npi);
      cprintf("Time between Pi/2 pulses: %f sec\r\n",TT);
      cprintf("Preselection offset frequency: %.0lf Hz\r\n",fpre);
      cprintf("Raman sequence start time: %.4lf ms\r\n",tram);
      cprintf("\r\n  Frequency Data (Hz):\r\n");
      cprintf("Int   3rd Pi/2     4th Pi/2     Cent Freq\r\n");
      for(ni=0; ni<nseq; ni++)
        cprintf("%2i  % 11.2lf  % 11.2lf  % 12.3lf\r\n",iref[ni],p2f3[ni],
                p2f4[ni],cf[ni]);

      for(ni=0; ni<nseq; ni++) {
/* compute mean, standard deviation, minimum, and maximum of data */
        ymean[ni] = 0.;
        ysig[ni] = 0.;
        ymin[ni] = amp[ni][0];
        ymax[ni] = amp[ni][0];
        for(i=0; i<=nstep; i++) {
          ymean[ni] += amp[ni][i];
          ysig[ni] += amp[ni][i]*amp[ni][i];
          if (amp[ni][i] < ymin[ni]) ymin[ni] = amp[ni][i];
          if (amp[ni][i] > ymax[ni]) ymax[ni] = amp[ni][i];
        }
        ymean[ni] /= (nstep + 1);
        ysig[ni] = sqrt((ysig[ni] - (nstep+1)*sqr(ymean[ni]))/nstep);
/*
Find "positive zero" crossings where curve passes through the mean.  Average
these together to get best initial guess for phase parameter in fit.
*/
        f1 = amp[ni][0] - ymean[ni];
        xpz = 0.0; nxpz = 0;
        xnz = 0.0; nxnz = 0;
        for(i=1; i<=nstep; i++) {
          f2 = amp[ni][i] - ymean[ni];
          if ((f1 < 0.0) != (f2 < 0.0)) {
            /* interpolate between the two points on either side of the
               mean to get the zero crossing */
            f1 = of[ni][i-1] - (of[ni][i]-of[ni][i-1]) * f1 / (f2-f1);
            f1 *= 2*Pi*TT;
            mod2Pi(&f1,0.0);
            if (f2 > 0.0) {
              xpz -= f1; nxpz++;
            } else {
              xnz -= f1; nxnz++;
            }
          }
          f1 = f2;
```

```
        }
        xpz /= nxpz;
        xnz = xnz / nxnz + Pi;
        mod2Pi(&xnz,0.0);
        xzero[ni] = (xpz+xnz)/2.0;
      }
      cprintf("\r\n  Fringe Statistics:\r\n");
      cprintf("Int   Mean    Std Dev      Min       Max   Zero(rad)\r\n");
      for(ni=0; ni<nseq; ni++)
        cprintf("%2i % 8.2E % 8.2E % 8.2E % 8.2E % 8.2E\r\n",iref[ni],
               ymean[ni],ysig[ni],ymin[ni],ymax[ni],xzero[ni],p2f3[ni],p2f4[ni]);
      cprintf("\r\nAny key to continue...\r\n");
      getautoc(&iac);

MainLoop:
      do {
        if (istext) {
          clrscr();
          textcolor(LIGHTGRAY);
          gotoxy(1,1);
          cprintf("%s has interferometers:",datfn);
          for(ni=0; ni<nseq; ni++) {
        if (ni == inti)
          textcolor(WHITE);
        else
          textcolor(LIGHTGRAY);
        cprintf("%2i",iref[ni]);
        if (donefit[ni])
          cprintf("f");
        else
          cprintf(" ");
          }
          textcolor(LIGHTGRAY);
          cprintf("\r\n  g  - graph data\r\n");
          cprintf("  f  - fit data\r\n");
          cprintf("  s  - save fit results\r\n");
          cprintf("1..4 - select interferometer\r\n");
          cprintf("  n  - next data file\r\n");
          cprintf("  a  - fit all interferometers, save results, goto next data file\r\n");
          cprintf("  A  - fit all data files, save results, quit\r\n");
          cprintf("  q  - quit\r\n");
        }
        switch(c = getautoc(&iac)) {
         case 'g':
          do {
            if (istext) {
              initgraph(&graphdriver, &graphmode,BGIDIR);
              if (graphresult() != grOk) {
                cprintf("fit: ERROR initializing graphics.\r\n");
                bye(1,nseq);
              }
              istext = FALSE;
              showfit = FALSE;
            }
            if (!donegraph) {
              donegraph = TRUE;
              ScrnX = getmaxx();
              ScrnY = getmaxy();
              NColor = getmaxcolor();
              winx1 = 50;
              winy1 = 30;
              winx2 = ScrnX - winx1;
              winy2 = ScrnY - winy1;
              fitwx2 = winx2 - 10;
              fitwx1 = fitwx2 - 140;
              fitwy1 = winy1 + 10;
              fitwy2 = fitwy1 + 50;
              setextrema(of[inti],amp[inti],nstep+1);
              changescale(&stepy,&inty,1);
            }
            cleardevice();
```

```
            drawgrid(xunits,yunits);
            plot(of[inti],amp[inti],nstep+1,badpt[inti],nbad[inti],YELLOW,DOT2);
            sprintf(s1,"Inteferometer #%1i",iref[inti]);    /* display plot title */
            settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            outtextxy((winx1+winx2)/2,winy1+10,s1);
            if (showfit) {
              fstepx = stepx*nxdiv / npts;
              for(i=0; i<=npts; i++) {
                xfit[i] = leftx + i*fstepx;
                myfunc((int)0,2*Pi*TT*xfit[i],parm[inti],&(yfit[i]),dydp,nparm);
              }
              plot(xfit,yfit,npts+1,badpt[inti],0,MAGENTA,LINE);
              setfillstyle(SOLID_FILL,BLACK);
              bar(fitwx1,fitwy1,fitwx2,fitwy2);
              setcolor(MAGENTA);
              rectangle(fitwx1,fitwy1,fitwx2,fitwy2);
              settextstyle(SMALL_FONT,HORIZ_DIR,0);
              settextjustify(CENTER_TEXT,RIGHT_TEXT);
              k = fitwy1;
              l = textheight(ffstr);
              outtextxy((fitwx1+fitwx2)/2,k,ffstr);
              k += 2;
              line(fitwx1,k+l,fitwx2,k+l);
              settextjustify(LEFT_TEXT,RIGHT_TEXT);
              for(i=1; i<=nparm; i++) {
                sprintf(s1,"%s:% 8.2E +-%7.1E",parmname[i],
                        parm[inti][i],psd[inti][i]);
                outtextxy(fitwx1+2,(k+=l),s1);
              }
            }
            switch (c=getch()) {
             case 'f':
              if (!showfit && donefit[inti])
                showfit = TRUE;
              else
                showfit = FALSE;
              break;
             case 0:
              switch(getch()) {
               case 75:
                leftx += stepx; break;
               case 77:
                leftx -= stepx; break;
               case 72:
                boty -= stepy; break;
               case 80:
                boty += stepy; break;
               case 115:
                changescale(&stepx,&intx,-1); break;
               case 116:
                changescale(&stepx,&intx,1); break;
               case 141:
                changescale(&stepy,&inty,1); break;
               case 145:
                changescale(&stepy,&inty,-1); break;
              }
              break;
             case '1': case '2': case '3': case '4':
              j = c - '1';
//*24         if ((hasint[j]>=0) && (hasint[j] != inti))
//*24           inti = hasint[j];
//*30b
              if (j < nseq)
                inti = j;
//*30e
              if (showfit && !donefit[inti]) showfit = FALSE;
              break;
             case 'q': case '\x1B':
              closegraph();
              istext = TRUE;
```

```
              break;
            }
          } while ((c != 'q') && (c != '\x1B'));
          c = 'g';
          break;
        case '1': case '2': case '3': case '4':
          j = c - '1';
//*24     if ((hasint[j]>=0) && (hasint[j] != inti))
//*24       inti = hasint[j];
//*30b
            if (j < nseq)
              inti = j;
//*30e
          break;
        case 's':
          if (donefit[0]||donefit[1]||donefit[2]||donefit[3]) {
            hi1 = hasint[0];
            hi2 = hasint[1];
            hi3 = hasint[2];
            hi4 = hasint[3];
            strcpy(outfn,datfn);
#if (docorr)
            strcpy(strchr(outfn,'.'),".fit");
#else
            strcpy(strchr(outfn,'.'),".ucf");
#endif
            if ((out_fp=fopen(outfn,"wt")) == NULL) {
              cprintf("fit - ERROR opening output file, '%s'\r\n",outfn);
              bye(1,nseq);
            }
            fprintf(out_fp,"'%s' has %i points per interferometer\n",
                    datfn,nstep+1);
            fprintf(out_fp,
            "  with T=%.4f ms, %i Pi pulses, Fpre=%.0lf Hz, Tram=%.4lf ms\n",
                    1000*TT,npi,fpre,tram);
            fprintf(out_fp,"Fit function: %s\n",ffstr);
            for (ni=0; ni<nseq; ni++)
              if (donefit[ni]) {
                fprintf(out_fp,"Interferometer %i fit with %i free parameters using %i points:\n",
                        iref[ni],nfpfit[ni],ndpfit[ni]);
                fprintf(out_fp,"center frequency =%13.4lf Hz:\n",cf[ni]);
                for(i=1; i<=nparm; i++)
                  fprintf(out_fp,"  %s = % 11.5E (+- %9.3E)\n",parmname[i],
                          parm[ni][i],psd[ni][i]);
                fprintf(out_fp,"  chi^2 = %9.3E\n",pchisq[ni]);
              }
            strcpy(s1,datfn);
            c_p = (char *)strchr(s1,'.');
            sprintf(c_p,"\0");
            c_p = ((c_p - s1) > 8) ? c_p-8 : s1;
            if (dohead) {
              for (ni=0; ni<nseq; ni++) nis[ni] = ni;
              /* *** insert sort function here *** so that nis[]
                 is a sort list of the intferometer numbers 1-4 */
              for(j=0; j<4; j++) {
                i = (j<nseq) ? iref[nis[j]] : 0;
                c = 'A' + j;
                fprintf(sp_fp,"A%c%1i,A%c%1isd,C%c%1i,C%c%1isd,Phi%c%1i,Phi%c%1isd,Chi%c%1i^2,",
                        c,i,c,i,c,i,c,i,c,i,c,i,c,i);
              }
              fprintf(sp_fp,"1-2,c1-2,1-2sd,3-4,c3-4,3-4sd,ar1/2,ar4/3,time(s),");
              fprintf(sp_fp,"TiltX(V),TiltY(V),%.11lf\n",(double)RECSHFT);
              fflush(sp_fp);
              dohead = FALSE;
            }
            fprintf(sp_fp,"%8s,%8.4f,%2i,%7.0lf,%8.4lf",
                    c_p,1000*TT,npi,fpre,tram);
            fprintf(sum_fp,"%8s, %8.4f, %2i, %7.0lf, %8.4lf:\n",
                    c_p,1000*TT,npi,fpre,tram);
//24*       for(i=0; i<4; i++) {
//24*         j = hasint[i];
```

```
//24*         if ((j >= 0) && donefit[j]) {
//30b
            for (ni=0; ni<4; ni++) {
              if (ni < nseq) {
                j = nis[ni];
//30e
                fprintf(sp_fp,",%10.4E,%9.3E,%4.1f,%3.1f,%11.5E,%8.2E,%8.2E",
                        parm[j][1],psd[j][1],100*parm[j][2],100*psd[j][2],
                        parm[j][3]*parm[j][4],
                        sqrt(sqr(psd[j][3])+sqr(psd[j][4])),pchisq[j]);
              } else {
                fprintf(sp_fp,",          ,         ,    ,   ,           ,        ,        ");
              }
            }
            for (ni=0; ni<nseq; ni++) {
              if (donefit[ni]) {
                fprintf(sum_fp,"  %1i ",iref[ni]);
                fprintf(sum_fp,
                  "(%4.1f,%3.1f) (% 8.2E,%7.1E) (%4.2f) %8.2E%4i:%1i\n",
                  100*parm[ni][2],100*psd[ni][2],parm[ni][4],psd[ni][4],
                  parm[ni][3],pchisq[ni],ndpfit[ni],nfpfit[ni]);
              }
            }
            f0 = (npi + 1) * RECSHFT;
            d1 = 2*Pi*TT;

            /* print frequency difference if we have both Interferometers 1 & 2 */
            bool1 = bool2 = FALSE;
            ar12 = -1.0; ar43 = -1.0;
//*24       if ((hi1>=0)&&(hi2>=0)&&(donefit[hi1])&&(donefit[hi2])&&
//*24           (parm[hi1][3]==parm[hi2][3])) {
//*25:
            if (dsf0&&(hi1>=0)&&(hi2>=0)&&(donefit[hi1])&&(donefit[hi2])&&
                (parm[hi1][3]==parm[hi2][3])) {
              ddf = ((double)parm[hi1][4]*parm[hi1][3] - parm[hi2][4]*parm[hi2][3]);
              mod2Pid(&ddf,0.0);
#if (docorr == 1)
              /* old method of correcting for frequency dependent delays
                 in the synthesizer's output filter: */
              rf1 = RFphase(p2f4[hi1]) - RFphase(p2f3[hi1]);
              rf2 = RFphase(p2f4[hi2]) - RFphase(p2f3[hi2]);
              cddf = ddf + 2*Pi*parm[hi1][3]*(rf1 - rf2);
#else
              cddf = 0.0;
#endif
              ar12 = parm[hi1][1] / parm[hi2][1];
              if (cf[hi2] < cf[hi1]) {
                ddf = -ddf;
                cddf = -cddf;
              }
              ddf /= d1;
              cddf /= d1;
              ddfs = sqrt(sqr(psd[hi1][4])+sqr(psd[hi2][4]))/d1;
              df = fabs(cf[hi1]-cf[hi2]);
//            fprintf(out_fp,
//              "1-2 freq diff in Hz (corrected): %.5lf (%.5lf) +- %9.3lE\n",
//              df+ddf,df+cddf,ddfs);
              fprintf(out_fp,
                "1-2 freq diff in Hz (corrected): %.5lf () +- %9.3lE\n",
                df+ddf,ddfs);
              ddf += df - f0;
              cddf += df - f0;
              fprintf(out_fp,
                "(1-2 freq diff - f0)/(N+1) in Hz (corrected): % 11.5lE () +- %9.3lE\n",
                ddf/(npi+1),ddfs/(npi+1));
              fprintf(sp_fp,",% 11.5lE, ,%9.3lE",ddf/(npi+1),ddfs/(npi+1));
              fprintf(sum_fp,"  (1-2 freq diff - f0)/(N+1) in Hz (corrected): % 11.5lE () +- %9.3lE\
n",
                      ddf/(npi+1),ddfs/(npi+1));
//            fprintf(sum_fp,"  amp ratio: %4.2f\n",ar12);
              bool1 = (fabs(df - f0)/(npi+1) > 0.003);
```

```
            } else {
              fprintf(sp_fp,",            ,            ,         ");
            }

            /* print frequency difference if we have both Interferometers 3 & 4 */
//*24       if ((hi3>=0)&&(hi4>=0)&&(donefit[hi3])&&(donefit[hi4])&&
//*24           (parm[hi3][3]==parm[hi4][3])) {
//*25:
            if (dsf0&&(hi3>=0)&&(hi4>=0)&&(donefit[hi3])&&(donefit[hi4])&&
                (parm[hi3][3]==parm[hi4][3])) {
              ddf = ((double)parm[hi3][4]*parm[hi3][3] - parm[hi4][4]*parm[hi4][3]);
              mod2Pid(&ddf,0.0);
#if (docorr == 1)
              /* old method of correcting for frequency dependent delays
                 in the synthesizer's output filter: */
              rf1 = RFphase(p2f4[hi3]) - RFphase(p2f3[hi3]);
              rf2 = RFphase(p2f4[hi4]) - RFphase(p2f3[hi4]);
              cddf = ddf + 2*Pi*parm[hi3][3]*(rf1 - rf2);
#else
              cddf = 0.0;
#endif
              ar43 = parm[hi4][1] / parm[hi3][1];
              if (cf[hi4] < cf[hi3]) {
                ddf = -ddf;
                cddf = -cddf;
              }
              ddf /= d1;
              cddf /= d1;
              ddfs = sqrt(sqr(psd[hi3][4])+sqr(psd[hi4][4]))/d1;
              df = fabs(cf[hi3]-cf[hi4]);
//            fprintf(out_fp,
//              "3-4 freq diff in Hz (corrected): %.5lf (%.5lf) +- %9.3lE\n",
//              df+ddf,df+cddf,ddfs);
              fprintf(out_fp,
                "3-4 freq diff in Hz (corrected): %.5lf () +- %9.3lE\n",
                df+ddf,ddfs);
              ddf += df - f0;
              cddf += df - f0;
              fprintf(out_fp,
                "(3-4 freq diff - f0)/(N+1) in Hz (corrected): % 11.5lE () +- %9.3lE\n",
                ddf/(npi+1),ddfs/(npi+1));
              fprintf(sp_fp,",% 11.5lE, ,%9.3lE",ddf/(npi+1),ddfs/(npi+1));
              fprintf(sum_fp,"  (3-4 freq diff - f0)/(N+1) in Hz (corrected): % 11.5lE () +- %9.3lE\
n",
                      ddf/(npi+1),ddfs/(npi+1));
//            fprintf(sum_fp,"  amp ratio: %4.2f\n",ar43);
              bool2 = (fabs(df - f0)/(npi+1) > 0.003);
            } else {
              fprintf(sp_fp,",            ,            ,         ");
            }
            if (ar12 < 0.0)
              fprintf(sp_fp,",    ");
            else
              fprintf(sp_fp,",%4.2f",ar12);
            if (ar43 < 0.0)
              fprintf(sp_fp,",    ");
            else
              fprintf(sp_fp,",%4.2f",ar43);
            fprintf(sp_fp,",%5lu,% 6.3f,% 6.3f\n",ltm,tltx,tlty);
//          fprintf(sp_fp,",%5lu,% 6.3f,% 6.3f,%2s\n",ltm,tltx,tlty,(iref[0]==1)?"12":"34");
            if (bool1) fprintf(sum_fp,"  WARNING: 1-2 center frequency separation does not match cal
culated recoil shift!\n");
            if (bool2) fprintf(sum_fp,"  WARNING: 3-4 center frequency separation does not match cal
culated recoil shift!\n");
            fclose(out_fp);
          }
          fflush(sum_fp);
          fflush(sp_fp);
          break;
        case 'f':
          clrscr();
```

```
          textcolor(LIGHTGRAY);
          parminit[1] = 1.0;
          parminit[2] = ysig[inti]/ymean[inti]*sqrt(2.0);
          parminit[3] = 1.0;
          parminit[4] = xzero[inti];
          intj = hasint[conj[iref[inti]-1]];
          bool1 = (intj>=0);
//*25b
          dosimfit = dsf0;
          if (dsf0) dosimfit = bool1;
//*25e
          if (dosimfit) {
            parminit[5] = 1.0;
            parminit[6] = ysig[intj]/ymean[intj]*sqrt(2.0);
            parminit[7] = xzero[intj];
          }
          gotoxy(1,1);
          cprintf("Fit function: %s\r\n",ffstr);
          cprintf("Parameter  Initial Value  Vary?\r\n");
          for (i=1; i<=nparm; i++) {
            cprintf("    %s          % 5.3f      %s\r\n",parmname[i],parminit[i],
                    varystr[pvinit[i]]);
          }
          gotoxy(1,locy[1][0]);
          cprintf("Simulaneously fit the conjugate interferometer?%s\r\n",
                  varystr[dosimfit]);
          cprintf("Display parameters after each fit iteration?   %s\r\n",
                  varystr[dispparm]);
          cprintf("Throw out all points more than X*sigma    (X = %3.2f)",
                  resfact);
          cprintf(" from the fitted curve.\r\n");
          cprintf("\r\n f  - fit with these initial parameters\r\n");
          cprintf("ESC - return to main menu\r\n");
          row = 0;
          col = 0;
          itable = 0;
PrintSimFit:
          if (dosimfit) {
            gotoxy(locx[2][0]-5,locy[2][0]-1);
            cprintf("(Conjugate Interferometer)");
            for(i=5; i<=7; i++) {
              j = (i==7) ? 3 : i-5;
              gotoxy(locx[2][0]-1,locy[2][j]);
              cprintf("% 5.3f      %s",parminit[i],varystr[pvinit[i]]);
            }
          } else {
            for(i=locx[2][0]-5,j=locy[2][0]-1; j<locy[2][0]+4; j++) {
              gotoxy(i,j);
              cprintf("                              ");
            }
          }
          do {
            gotoxy(locx[itable][col],locy[itable][row]);
            textcolor(YELLOW);
            switch(c = getautoc(&iac)) {
             case '0': case '1': case '2': case '3': case '4': case '5':
             case '6': case '7': case '8': case '9': case '.': case '-':
             case '+':
              if ((((itable==0)||(itable==2))&&(col == 0)) ||
                  ((itable==1)&&(row==2))) {
                putch(c);
                s1[0] = c;
                for (i=1; (s1[i]=getche()) != '\r'; i++);
                s1[i] = '\0';
                if (sscanf(s1,"%f\n",&pval)) {
                  if (itable == 1) {
                    if (pval < 0.0) pval = 0.0;
                    resfact = pval;
                    gotoxy(locx[itable][col],locy[itable][row]);
                    cprintf("%3.2f",resfact);
                  } else {
```

```
              parminit[row+1] = pval;
              gotoxy(locx[itable][col]-1,locy[itable][row]);
              cprintf("% 5.3f",parminit[row+1]);
            }
          } else {
            textcolor(LIGHTGRAY);
            gotoxy(locx[itable][col]-1,locy[itable][row]);
            if (itable == 1)
              cprintf("%3.2f",resfact);
            else
              cprintf("% 5.3f",parminit[row+1]);
            textcolor(YELLOW);
          }
        }
        break;
       case 'y':
        if (((itable == 0) || (itable == 2)) && (col == 1)) {
          pvinit[row+1] = YES;
          gotoxy(locx[itable][col],locy[itable][row]);
          cprintf("%s",varystr[YES]);
        } else if ((itable == 1) && (col == 0)) {
          gotoxy(locx[itable][col],locy[itable][row]);
          cprintf("%s",varystr[YES]);
          if (row == 0) {
            dosimfit = bool1;
            goto PrintSimFit;
          } else if (row == 1) {
            dispparm = TRUE;
          }
        }
        break;
       case 'n':
        if (((itable == 0) || (itable == 2)) && (col == 1)) {
          pvinit[row+1] = NO;
          gotoxy(locx[itable][col],locy[itable][row]);
          cprintf("%s",varystr[NO]);
        } else if ((itable == 1) && (col == 0)) {
          gotoxy(locx[itable][col],locy[itable][row]);
          cprintf("%s",varystr[NO]);
          if (row == 0) {
            dosimfit = FALSE;
            goto PrintSimFit;
          } else if (row == 1) {
            dispparm = FALSE;
          }
        }
        break;
       case 'a':
        if (((itable == 0) || (itable == 2)) && (col == 1)) {
          pvinit[row+1] = 2;
          gotoxy(locx[itable][col],locy[itable][row]);
          cprintf("%s",varystr[2]);
        }
        break;
       case '\r':
        tablemove(itable,1,0,&row,&col); break;
       case '\t':
        row = 0;
        col = 0;
        if (++itable > dosimfit+1) itable = 0;
        break;
       case 0:
        switch(getch()) {
         case 75:
          tablemove(itable,0,-1,&row,&col); break;
         case 77:
          tablemove(itable,0,1,&row,&col); break;
         case 72:
          tablemove(itable,-1,0,&row,&col); break;
         case 80:
          tablemove(itable,1,0,&row,&col); break;
```

```
          }
          break;
         case 'q': case 0x1B:                    /* ESC key */
          goto MainLoop;
        }
      } while (c != 'f');
      clrscr();
      textcolor(LIGHTGRAY);
      /* assign amplitude scaling factor */
      if (dosimfit)
        yscale = (ymean[inti] + ymean[intj])/2;
      else
        yscale = ymean[inti];
      /* check for negative amplitudes in 1st interferometer */
      if (ymin[inti] < 0.0) {
        yoff1 = -1.1*ymin[inti];
        cprintf("Warning!  Some amplitude values of Inteferometer %i are negative.\r\n",
                iref[inti]);
        beep(1);
        cprintf("  - shifting all values up by %8.2E\r\n",yoff1);
      } else
        yoff1 = 0.0;
      /* scale and shift data from 1st interferometer, if necessary */
      for(i=0,ndata=0; i<=nstep; i++,ndata++) {
        xdata[ndata+1] = 2*Pi*TT*of[inti][i];
        ydata[ndata+1] = (amp[inti][i] + yoff1) / yscale;   /* scale data */
        sdata[ndata+1] = 1.0;
      }
      idsref[0] = ndata + 1;
      nparm = 4;
      if (dosimfit) {
        /* check for negative amplitudes in conjugate interferometer */
        if (ymin[intj] < 0.0) {
          yoff2 = -1.1*ymin[intj];
          cprintf("Warning!  Some amplitude values of Inteferometer %i are negative.\r\n",
                  iref[inti]);
          beep(1);
          cprintf("  - shifting all values up by %8.2E\r\n",yoff2);
        } else
          yoff2 = 0.0;
        /* scale and shift data from conjugate interferometer, if necessary */
        for(i=0; i<=nstep; i++,ndata++) {
          xdata[ndata+1] = 2*Pi*TT*of[intj][i];
          ydata[ndata+1] = (amp[intj][i] - yoff2) / yscale;   /* scale data */
          sdata[ndata+1] = 1.0;
        }
        idsref[1] = ndata + 1;
        nparm = 7;
      }
      for(i=1,n2pass=0; i<=nparm; i++) {
        /* assign initial parameter values */
        fitparm[i] = parminit[i];
        pvary[i] = pvinit[i];
        /* look for any parameters set to "AutoVary" */
        if (pvinit[i] == AUTO) {
            pvary[i] = NO;
            n2pass++;
        }
      }
      /* allocate temporary scratch space for fit routines */
      covar = matrix((long)1,(long)nparm,(long)1,(long)nparm);
      alpha = matrix((long)1,(long)nparm,(long)1,(long)nparm);

      chisq = RunFit(dispparm,ndata,xdata,ydata,sdata,nparm,fitparm,
                     pvary,dosimfit,idsref,yscale,covar,alpha);

      /* if there are any "AutoVary" parameters, run fit again */
      if (n2pass) {
        for(i=1; i<=nparm; i++) {
          /* save old parameter values */
          oldparm[i] = fitparm[i];
```

```
          oldvar[i] = covar[i][i];
          /* turn on all "AutoVary" parameters */
          if (pvinit[i] == AUTO)
            pvary[i] = YES;
        }
        if (dispparm) clrscr();
        cprintf("Running fit again:\r\n");
        cprintf("  - using the parameters resulting from the first pass\r\n");
        cprintf("  - this time also varying all parameters set to \"Auto\"\r\n");
        newchisq = RunFit(dispparm,ndata,xdata,ydata,sdata,nparm,fitparm,
                          pvary,dosimfit,idsref,yscale,covar,alpha);
        if (dispparm)
          clrscr();
        else
          printf("\n");
        if (newchisq > chisq) {
          cprintf("New chi-squared worse (larger) than previous value.\r\n");
          cprintf(" -> keep OLD parameters\r\n");
          for(i=1; i<=nparm; i++) {
            fitparm[i] = oldparm[i];
            covar[i][i] = oldvar[i];
            if (pvinit[i] == AUTO) pvary[i] = NO;
          }
        } else {
          cprintf("New chi-squared better (smaller) than previous value.\r\n");
          cprintf(" -> use NEW parameters\r\n");
          chisq = newchisq;
        }
      }

      if (resfact > 0.0) {
      /* compute residuals and standard deviation of residuals */
        cprintf("\r\nComputing residuals...\r\n");
        resmean = 0.0;
        ressd = 0.0;
        for(j=0,i=1; j<=dosimfit; j++)
          for(; i<idsref[j]; i++) {
            myfunc(j,xdata[i],fitparm,&f1,dydp,nparm);
            res[i] = ydata[i] - f1;
            resmean += res[i];
            ressd += res[i]*res[i];
          }
        resmean /= ndata;
/*      ressd = sqrt(ressd/ndata - resmean*resmean); */
        ressd = sqrt(ressd/(ndata-1));
        /* throw out any points whose residual is more than "resfact"
           standard deviations ("ressd") away from "resmean" */
        cprintf("Throwing out all points farther than %3.2f*sigma away from fitted curve...\r\n"
,
                resfact);
        for(j=0,i=1,k=1,l=0,nignore=0; j<=dosimfit; j++) {
          l = (j==0) ? inti : intj;
          nbad[l] = 0;
          for(; i<idsref[j]; i++,k++) {
            xdata[k] = xdata[i];
            ydata[k] = ydata[i];
/*          if (fabs(res[i]-resmean) >= resfact*ressd) { */
            if (fabs(res[i]) >= resfact*ressd) {
              k--;
              nignore++;
              badpt[l][nbad[l]++] = (j==0)? i-1 : i-idsref[0];
            }
          }
          cprintf("  %3i points from Interferometer %i.\r\n",nbad[l],
                  iref[l]);
          ndata -= nbad[l];
        }
        idsref[0] -= nbad[inti];
        if (dosimfit) idsref[1] -= nignore;
        /* if points were thrown out, refit using the remaining points */
        if (nignore > 0)
```

```
          chisq = RunFit(dispparm,ndata,xdata,ydata,sdata,nparm,fitparm,
                         pvary,dosimfit,idsref,yscale,covar,alpha);
      }

      /* store fit results and clean up */
      for(i=1; i<=4; i++) {
        parm[inti][i] = fitparm[i];
        psd[inti][i] = sqrt(chisq*covar[i][i]);
      }
      /* return to original scale: */
      parm[inti][1] *= yscale;
      psd[inti][1] *= yscale;
      pchisq[inti] = chisq * sqr(yscale);
      if (dosimfit) {
        parm[intj][1] = fitparm[5] * yscale;
        psd[intj][1] = sqrt(chisq*covar[5][5]) * yscale;
        parm[intj][2] = fitparm[6];
        psd[intj][2] = sqrt(chisq*covar[6][6]);
        parm[intj][3] = parm[inti][3];
        psd[intj][3] = psd[inti][3];
        parm[intj][4] = fitparm[7];
        psd[intj][4] = sqrt(chisq*covar[7][7]);
        pchisq[intj] = pchisq[inti];
      }
      cprintf("\r\nFinal:  chi^2 = %9.3E\r\n",pchisq[inti]);
      standardform(parm[inti],psd[inti]);
      if (dosimfit) standardform(parm[intj],psd[intj]);
      for(i=1; i<=4; i++) {
        cprintf("  %s = % 11.5E (+- %9.3E)",parmname[i],parm[inti][i],
                psd[inti][i]);
        if (dosimfit)
          cprintf("  |  % 11.5E (+- %9.3E)\r\n",parm[intj][i],psd[intj][i]);
        else
          printf("\n");
      }
      free_matrix(covar,(long)1,(long)nparm,(long)1,(long)nparm);
      free_matrix(alpha,(long)1,(long)nparm,(long)1,(long)nparm);
      donefit[inti] = TRUE;
      ndpfit[inti] = ndata;
      for(i=1,nfpfit[inti]=0; i<=nparm; i++)
        if (pvary[i]) nfpfit[inti]++;
      if (dosimfit) {
        donefit[intj] = TRUE;
        ndpfit[intj] = ndpfit[inti];
        nfpfit[intj] = nfpfit[inti];
        nparm = 4;
      }
      cprintf("\nAny key to return to main menu\r\n");
      getautoc(&iac);
      break;
     case 'q':
      if (!istext) closegraph();
      break;
     case 'a': case 'A':
        autocommand[0] = '\0';
        for (ni=0; ni<nseq; ni++) {
          j = hasint[conj[iref[ni]-1]];
//*24     if ((j < 0) || (j > ni)) {
//*24       sprintf(s1,"%1iff ",iref[ni]);
//*25b

          if ((j < 0) || (j > ni) || !dsf0) {
            sprintf(s1,"%1iff ",ni+1);
//*25e
            strcat(autocommand,s1);
          }
        }
        strcat(autocommand,"sn");
        if (c == 'A') strcat(autocommand," A");
        nautocommand = strlen(autocommand);
        iac = 1;
        isauto = TRUE;
```

```
          break;
        }
      } while ((c != 'q') && (c != 'n'));
      FreeDynam(nseq);
    } else {
      c = 'q';
    }
  } while (c != 'q');
  fclose(sum_fp);
  fclose(sp_fp);
  return 0;
}


int fgetline(FILE *f_fp, char *line, int *l)
/*
  Read up to "l" characters from stream "f_fp" and store them in
"line[]".  Stop if a '\n' or EOF is reached.
  Returns:
    TRUE, if "l" chars are read without encountering a '\n' or EOF
    FALSE, otherwise
*/
{
  char c;
  int i = 0;

  if (*l == 0) return TRUE;
  while(((c=fgetc(f_fp)) != '\n') && (c != EOF) && (i < *l)) {
    line[i++] = c;
  }
  if ((i==*l) && (c!='\n') && (c!=EOF)) {
    line[i-1] = '\0';
    return TRUE;
  }
  line[i] = '\0';
  *l = i;
  return FALSE;
}


void myfunc(int ids, float x,float *parm,float *y,float *dydp,int nparm)
{
  int i;
  float phi1,phi2,fc,fs;

  if (ids == 0) {
/* *y = parm[1]*(1 + parm[2]*sin(parm[3]*(x+parm[4]))); */
    phi1 = x + parm[4];
    phi2 = parm[3]*phi1;
    fs = sin(phi2);
    fc = parm[1]*parm[2]*cos(phi2);
    dydp[1] = 1.0 + parm[2]*fs;
    *y = parm[1]*dydp[1];
    dydp[2] = parm[1]*fs;
    dydp[3] = fc*phi1;
    dydp[4] = fc*parm[3];
    for(i=5; i<=nparm; i++)
      dydp[i] = 0.0;
  } else if (ids == 1) {
/* *y = parm[5]*(1 + parm[6]*sin(parm[3]*(x+parm[7]))); */
    phi1 = x + parm[7];
    phi2 = parm[3]*phi1;
    fs = sin(phi2);
    fc = parm[5]*parm[6]*cos(phi2);
    dydp[5] = 1.0 + parm[6]*fs;
    *y = parm[5]*dydp[5];
    dydp[6] = parm[5]*fs;
    dydp[3] = fc*phi1;
    dydp[7] = fc*parm[3];
    dydp[1] = 0.0;
    dydp[2] = 0.0;
```

```
    dydp[4] = 0.0;
  }

/*
  *y = parm[1] + parm[2]*sin(parm[3]*x+parm[4]);
  dydp[1] = 1.0;
  dydp[2] = sin(parm[3]*x+parm[4]);
  dydp[3] = parm[2]*x*cos(parm[3]*x+parm[4]);
  dydp[4] = parm[2]*cos(parm[3]*x+parm[4]);
*/
/*
  *y = parm[1]*(1 + parm[2]*sin((1+parm[3])*(x+parm[4])));
  dydp[1] = 1 + parm[2]*sin((1+parm[3])*(x+parm[4]));
  dydp[2] = parm[1]*sin((1+parm[3])*(x+parm[4]));
  dydp[3] = parm[1]*parm[2]*(x+parm[4])*cos((1+parm[3])*(x+parm[4]));
  dydp[4] = parm[1]*parm[2]*(1+parm[3])*cos((1+parm[3])*(x+parm[4]));
*/
}


void autoscale(float min, float max, int ndiv, float *minscaled, float *step, float *intscaled)
{
  float a,b,f;

  f = log10((max - min)/ndiv);
  a = floor(f);
  b = f - a;
  a = pow(10,a);
  b = pow(10,b);
  if (b > 5.1) *step = a*10;
  else if (b > 2.1) *step = a*5;
  else if (b > 1.1) *step = a*2;
  else *step = a;
  *minscaled = floor(min/(*step)) * *step;
  *intscaled = *step * ndiv;
}


void changescale(float *step, float *interval, int dir)
{
  float s,b,f;

  if (dir != 0) {
    f = log10(*step);
    b = f - floor(f);
    b = pow(10,b);
    if (((dir > 0) && (fabs(b-2) < 0.00001)) ||
        ((dir < 0) && (fabs(b-5) < 0.00001))) s = 2.5;
    else s = 2.0;
    if (dir > 0) {
      *step *= s;
      *interval *= s;
      changescale(step,interval,dir-1);
    } else {
      *step /= s;
      *interval /= s;
      changescale(step,interval,dir+1);
    }
  }
}


void rtog(float x, float y, int *gx, int *gy)
{
  *gx = (x-leftx)/intx*(winx2-winx1) + winx1;
  *gy = winy2 - (y-boty)/inty*(winy2-winy1);
}


void mydot(int gx, int gy, int type)
{
```

```
  switch(type) {
   case TICKX1:
     line(gx-2,gy,gx+2,gy);
     break;
   case TICKY1:
     line(gx,gy-2,gx,gy+2);
     break;
   case DOT2:
     fillellipse(gx,gy,2,2);
     break;
   case DOT1:
     fillellipse(gx,gy,1,1);
     break;
   case CROSS:
     line(gx-4,gy-4,gx+4,gy+4);
     line(gx-4,gy+4,gx+4,gy-4);
     break;
   case LINE:
     lineto(gx,gy);
     break;
   case POINT:
   default:
     putpixel(gx,gy,getcolor());
     break;
  }
}


void setextrema(float *x, float *y, int n)
{
  float minx,maxx, miny,maxy;   /* extrema of the data */
  int i;

  minx = maxx = x[0];
  miny = maxy = y[0];
  for(i=1; i<n; i++) {
    if (x[i] > maxx) maxx = x[i];
    if (x[i] < minx) minx = x[i];
    if (y[i] > maxy) maxy = y[i];
    if (y[i] < miny) miny = y[i];
  }
  autoscale(minx,maxx,nxdiv,&leftx,&stepx,&intx);
  autoscale(miny,maxy,nydiv,&boty,&stepy,&inty);
}


void drawgrid(char *xunits, char *yunits)
{
 int gx1,gy1,gleftx,gboty,grightx,gtopy;
 int gx[100],gy[100];
 int mxnsd;
 int i,j;
 char s1[80];

 int
   font = DEFAULT_FONT,
   fontsize = 1;

  setcolor(LIGHTGRAY);
  rtog(leftx,boty,&gleftx,&gboty);
  rtog(leftx+stepx*nxdiv,boty+stepy*nydiv,&grightx,&gtopy);
  mxnsd = max(nxdiv*nxsubdiv,nydiv*nysubdiv);
  for(i=0; i<mxnsd; i++)
    rtog(leftx+i*stepx/nxsubdiv,boty+i*stepy/nysubdiv,&(gx[i]),&(gy[i]));
  for(i=0; i<=nxdiv; i++) {
    rtog(leftx+i*stepx,boty,&gx1,&gy1);
    line(gx1,gboty,gx1,gtopy);
    for(j=0; j<(nydiv*nysubdiv); j++)
      mydot(gx1,gy[j],TICKX1);
  }
  for(i=0; i<=nydiv; i++) {
```

```
    rtog(leftx,boty+i*stepy,&gx1,&gy1);
    line(gleftx,gy1,grightx,gy1);
    for(j=0; j<(nxdiv*nxsubdiv); j++)
      mydot(gx[j],gy1,TICKY1);
  }
  sprintf(s1,"%7.1E %s",leftx,xunits);
  settextstyle(font,HORIZ_DIR,fontsize);
  settextjustify(LEFT_TEXT,RIGHT_TEXT);
  outtextxy(gleftx,gboty+4,s1);
  sprintf(s1,"%7.1E",leftx+stepx*nxdiv);
  settextstyle(font,HORIZ_DIR,fontsize);
  settextjustify(CENTER_TEXT,RIGHT_TEXT);
  outtextxy(grightx,gboty+4,s1);
  sprintf(s1,"%6.0E per div",stepx);
  settextstyle(font,HORIZ_DIR,fontsize);
  settextjustify(CENTER_TEXT,RIGHT_TEXT);
  rtog(leftx+nxdiv/2*stepx,boty,&gx1,&gy1);
  outtextxy(gx1,gboty+4,s1);
  sprintf(s1,"%7.1E %s",boty,yunits);
  settextstyle(font,VERT_DIR,fontsize);
  settextjustify(RIGHT_TEXT,LEFT_TEXT);
  outtextxy(gleftx-5,gboty,s1);
  sprintf(s1,"%7.1E",boty+stepy*nydiv);
  settextstyle(font,VERT_DIR,fontsize);
  settextjustify(RIGHT_TEXT,CENTER_TEXT);
  outtextxy(gleftx-5,gtopy,s1);
  sprintf(s1,"%6.0E per div",stepy);
  settextstyle(font,VERT_DIR,fontsize);
  settextjustify(RIGHT_TEXT,CENTER_TEXT);
  rtog(leftx,boty+nydiv/2*stepy,&gx1,&gy1);
  outtextxy(gleftx-5,gy1,s1);
}


void plot(float *x, float *y, int n, int *bp, int nb, int color, int dot)
{
  int gx1, gy1;
  int i,ib=0;

  for(i=0; i<n; i++) {
    rtog(x[i],y[i],&gx1,&gy1);
    setcolor(color);
    setfillstyle(SOLID_FILL,color);
    if (dot == LINE) {
      if (i==0) moveto(gx1,gy1);
      mydot(gx1,gy1,dot);
    } else {
      if ((nb > 0) && (i == bp[ib])) {
        ib++;
        setcolor(LIGHTRED);
        mydot(gx1,gy1,CROSS);
        setcolor(color);
      } else
        mydot(gx1,gy1,dot);
    }
  }
}


void dataerr(int l)
{
  cprintf("fit: ERROR in data file, line %i.\r\n",l);
  fclose(dat_fp);
  bye(1,0);
}


void bye(int err, int ns)
{
  /* if variable space was allocated, free it */
  if (ns > 0) FreeDynam(ns);
```

```
  /* close files */
  fclose(sum_fp);
  fclose(sp_fp);
  exit(err);
}


void FreeDynam(int ns)
{
  int i;

  free(xdata+1);         /* starting index = 1 */
  free(ydata+1);         /* starting index = 1 */
  free(sdata+1);         /* starting index = 1 */
  free(res+1);           /* starting index = 1 */
  for(i=0; i<ns; i++) {
    free(of[i]);
    free(amp[i]);
    free(badpt[i]);
    free(parm[i]+1);     /* starting index = 1 */
    free(psd[i]+1);      /* starting index = 1 */
  }
  free(of);
  free(amp);
  free(badpt);
  free(parm);
  free(psd);
}


void tablemove(int it, int dr, int dc, int *r, int *c)
{
  *r += dr;
  if (*r < 0) {
    if (*c == 0) {
      *r = nrow[it] - 1;
      *c = ncol[it] - 1;
    } else {
      *r += nrow[it];
      tablemove(it,0,-1,r,c);
    }
  } else if (*r >= nrow[it]) {
    if (*c >= (ncol[it]-1)) {
      *r = 0;
      *c = 0;
    } else {
      *r -= nrow[it];
      tablemove(it,0,1,r,c);
    }
  }

  *c += dc;
  if (*c < 0) {
    if (*r == 0) {
      *r = nrow[it] - 1;
      *c = ncol[it] - 1;
    } else {
      *c += ncol[it];
      tablemove(it,-1,0,r,c);
    }
  } else if (*c >= ncol[it]) {
    if (*r >= (nrow[it]-1)) {
      *r = 0;
      *c = 0;
    } else {
      *c -= ncol[it];
      tablemove(it,1,0,r,c);
    }
  }
}
```

```
int mod2Pi(float *phi,float cent)
{
  float
   oldphi = *phi,
   p1 = cent - Pi,
   p2 = cent + Pi,
   twoPi = 2*Pi;

  while (*phi > p2) *phi -= twoPi;
  while (*phi < p1) *phi += twoPi;
  return (!(oldphi == *phi));
}


int mod2Pid(double *phi,double cent)
{
  double
   oldphi = *phi,
   p1 = cent - Pi,
   p2 = cent + Pi,
   twoPi = 2*Pi;

  while (*phi > p2) *phi -= twoPi;
  while (*phi < p1) *phi += twoPi;
  return (!(oldphi == *phi));
}


char getautoc(int *i)
{
  char c;

  if (isauto) {
    if (*i >= nautocommand) isauto = FALSE;
    if (*i <= nautocommand) {
      c = autocommand[*i-1];
      (*i)++;
      return c;
    }
  }
  return getch();
}


void beep(int i)
{
 int
  f1, f2,        /* frequency of beep (Hz) */
  d1, d2;        /* duration of beep (ms) */

  switch(i) {
   case 2:
    f1 = 1000;
    d1 = 50;
    f2 = 1000;
    d2 = 30;
    break;
   default:
    f1 = 1000;
    d1 = 100;
    f2 = 0;
    d2 = 0;
  }
  sound(f1);
  delay(d1);
  nosound();
  delay(d2);
  sound(f2);
  delay(d1);
  nosound();
```

```
}


void standardform(float p[], float psd[])
/* put parameters in "standard" form */
{
  float p1;

  if (p[3] < 0) {
    p[3] = -p[3];
    p[2] = -p[2];
  }
  if (p[2] < 0) {
    p[2] = -p[2];
    p[4] += Pi/p[3];
  }
  p1 = p[4] * p[3];
  if (mod2Pi(&p1,0.0)) {
    beep(1);
    cprintf("WARNING: parameter shifted by a multiple of 2Pi\r\n");
  }
  p[4] = p1 / p[3];
/*
  p1 = psd[4] * p[3];
  if (mod2Pi(&p1,0.0)) {
    beep(2);
    cprintf("WARNING: standard deviation of parameter shifted by a multiple of 2Pi\r\n");
    getch();
  }
  psd[4] = p1 / p[3];
*/
}


float RunFit(int dodisp,
             int n, float x[], float y[], float s[],
             int nparm, float p[], int pvary[],
             int dosimfit, int idsref[], float yscale,
             float **covar, float **alpha)
{
 int
  i,
  nfit=0,                        /* number of parameters that vary */
  iter;
 float
  chisq = 0.0,
  alamda = -1.0;

  /* count the number of free parameters */
  for(i=1; i<=nparm; i++)
    if (pvary[i]) nfit++;

  if (dodisp) {
    cprintf("Fit %i free parameters to %s using %i points:\r\n",
            nfit,ffstr,n);
    cprintf("Iter    A      B      C      D");
    if (dosimfit) cprintf("    |    A      B      D");
    cprintf("        chi^2     lamda\r\n");
  } else
    cprintf("Fitting %i free parameters using %i points",nfit,n);
    /*
    void mrqmin(float x[], float y[], float sig[], int n, float a[], int ia[],
        int ma, float **covar, float **alpha, float *chisq,
        void (*funcs)(float, float [], float *, float [], int), float *alamda)
    */
  for(iter=0; (iter<=100) && (alamda<=1000.); iter++) {
    if (dodisp) {
      cprintf("%3i: % 5.3f % 5.3f % 5.3f % 5.3f",iter,p[1]*yscale,
              p[2],p[3],p[4]);
      if (dosimfit)
        cprintf("  | % 5.3f % 5.3f % 5.3f",p[5]*yscale,p[6],
```

```
              p[7]);
        cprintf("    %9.3E  % 6.0E\r\n",chisq/(n-nfit)*yscale*yscale,
              alamda);
    } else {
        cprintf(".");
    }
    mymrqmin(x,y,s,n,idsref,p,pvary,nparm,covar,
             alpha,&chisq,myfunc,&alamda);
  };
  printf("\n");
  alamda = 0.0;
  mymrqmin(x,y,s,n,idsref,p,pvary,nparm,covar,
           alpha,&chisq,myfunc,&alamda);
  if (dodisp) {
    cprintf("Press any key to continue.\r\n");
    getch();
  }
  return chisq/(n-nfit);
}


unsigned long TimeToSeconds(char *s)
/*
  Convert time in hh:mm:ss form stored in string "s" to seconds from
00:00:00.
*/
{
  char
   *c1_p = NULL,
   *c2_p = NULL;
  unsigned long ul;

  if (((c1_p = strchr(s,':')) < s+2) || ((c2_p = strchr(c1_p+1,':')) < s+5))
    return (unsigned long)0;
  ul = (unsigned long)3600*atoi(s);
  ul += (unsigned long)60*atoi(c1_p+1);
  ul += (unsigned long)atoi(c2_p+1);
  return ul;
}
```

C.3 DigFil.C

The DigFil.C code compiled in *Borland* Turbo C++ 3.0 and running under DOS controls the active part of the vibration isolation system discuss in Section 3.4. It installs a small memory resident routine that traps a hardware interrupt generated every time the analog-to-digital (AD) pc-board inside the computer samples another value from the accelerometer. Using a discrete transform with operator controllable parameters, it transforms this digital input signal in real time into an output control value which is converted back to an analog signal by the same pc-board. This analog control signal goes eventually to the solenoid coil which pushes against the freely moving part of the VI tower (see Figure 3.15) to counteract the motion detected by the accelerometer. A particular important feature of the code is the ability to change the parameters that define the discrete transform *while* the feedback loop is closed. This feature allows the feedback parameters to be optimized without opening and re-closing the loop, thus minimizing the time spent waiting for feedback transients to die away.

```c
/*
  DIGFIL.C                                Joel M. Hensley
Has the National Intruments AT-MIO-16XE-50 board perform as an "almost"
real-time digital controller for a vibration isolation system.  Using
interrupts digital values are read in from the board, filtered, and
then given directly back to the board for conversion to an analog
output signal.

2.0     9/95
  Revised to run the AT-MIO-16X board with 16-bit resolution.
2.1     9/15/95
  Added feature to allow the operator to vary the gain of the monitor
output (DAC1).
  Added feature to automatically decrease the gain when the control
output exceeds a certain threshold (THRAIL) and increase the gain when
it settles below another threshold (THMIN).
2.3     10/16/95
  Allow operator to remove high-pass filter from compensation function.
  Added feature to save output values of internal transfer function
components into a buffer so that they can be viewed later.
2.4     12/12/95
  Add two more compensators.
2.5     1/15/95
  Add a gate to the buffer save feature so that we can store values of
the error signal at specific points in time.  The gate signal will come
in on bipolar channel AI1/9.
3.0     3/9/96
  Same as version 2.5 but drives AT-MIO-16XE-50 board instead of AT-MIO-16X
3.1     4/17/96
  Change format of input file and internal definition of transfer functions
to increase flexibility.
3.2     3/27/97
  Fix bug that incorrectly assigned the interupt number.
*/
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#include "mynidefs.h"
#include "arrays.h"


enum {FALSE,TRUE};

/* conversion factor for CMG3 (serial number 3133) in V/(mm/s*s) */
#define ACCELCF   (0.995)
/* extrema values for bipolar 16-bit conversion: */
#define MAXVAL  32767           /* maximum allowed digital value */
#define MINVAL  -32768          /* minimum allowed digital value */
/*
The following parameters are used only if they are not set correctly
in the input file: */
#define EXTCONT   0     /* sample rate controlled (1=external,0=internal) */
#define SAMPRATE  4000  /* sample rate (points / sec) */
                        /*  - maximum from command line: 8000
                            - maximum from Turbo C: ~4000 */
#define NDITHAVG  4     /* # of past samples to avg over */
#define MODFACT   1.1892
#define THMAX     0.9   /* threshold where gain is automatically lowered */
#define THMIN     0.1   /* threshold where gain is automatically raised */
        /* thresholds defined in terms of percentage of maximum range */
#define ONTHRESH  1     /* threshold for turn-on comparator */

#define INFILE    "digfil.in"
#define INFVER    2
#define OUTFILE   "digfil.out"

#define NTF             4       /* max # of definable transfer functions */
```

```c
#define NELEMMAX        10      /* max # of definable elements per TF */
#define NSBUF           5       /* number of save buffers */
#define SBUFSIZ         1000    /* size of save buffers */
#define GBUFSIZ         10000   /* size of gate save buffer */
#define BGMASK          1       /* gate comes in on Digital Port A0 */
#define NTGSTEP   28    /* number of steps before achieving final gain */
                        /* modifcation factor for variable compensator */
enum {UNDERRAIL,NORAIL,OVERRAIL};
const int printlevel = 1;


unsigned int
  igate = 0;            /* number of external gate events */
int
  ndithavg=NDITHAVG,    /* number of past samples to average over */
  dithfact=NDITHAVG,    /* multiplicative factor used to increase
                           resolution after dither averaging */
  onthresh=ONTHRESH,    /* threshold for turn-on comparator */
  tfnelem[NTF],         /* number of elements in each TF */
  **tfes,               /* individual elements enabled? */
  **tfe,                /* element type:
                             0 = HP
                             1 = LP
                             2 = CU
                             3 = CD */
  **tfepos,             /* element index for each position */
  **tfpose,             /* element position for each element index */
  nparme[4] = {2,2,3,3},        /* number of parameters in element type */
  Atf = 0,              /* output A computed from this TF */
  Btf = 1,              /* output B computed from this TF */
  tfreq = -1,           /* TF requested */
  TOtf = 3,             /* vel=0 computed from this TF */
  bab = 1,              /* 0: output A inactive, output B active
                           1: output A active, ouput B inactive */
  israiled,             /* control output at or past rail?
                            UNDERAIL = abs(control out) < thmin
                            NORAIL = thmin < abs(control out) < thmax
                            OVERRAIL = abs(control out) > thmax */
  itempgain = 0,                /* =log(tempfact)/log(modfact) */
  cstate = 0,           /* 0 = FB loop open
                           1 = FB closed, no pending messages
                           2 = waiting for vel=0 to close FB loop
                           3 = display "FB loop closed" message
                           4 = waiting for vel=0 to make change in TF
                           5 = display "TF changed" message
                           6 = wait for vel=0 to change active TF
                           7 = display "Active TF changed" message */
  monab = 1,            /* 0: monitor output B,
                           1: monitor output A */
  monpos = 0,           /* select which TF element is monitored at DAC1 */
  imog = 2,             /* index for monitor output gain */
  buffsel = -1,         /* selects the buffer where values are stored:
                            -1  = no storage
                            0-4 = buffer 1-5 */
  isbuff[NSBUF],        /* buffer index */
  dogatebuf = FALSE,    /* enable gated save buffer */
  gberr = 0,            /* 0 = no error,
                           1 = buffer full, message not displayed,
                           2 = buffer full, message displayed,
                           3 = sequence of 4 gates were not
                               within 200ms of each other */
  gbarm = 3,            /* 3 = wait for rising edge of preselection pulse
                           2 = wait for falling edge of preselection pulse
                           1 = wait for rising edge of a Pi/2 pulse
                           0 = wait for falling edge of a Pi/2 pulse */
  savebuff[NSBUF][SBUFSIZ],     /* buffer storage space */
  donebuff[NSBUF],      /* buffer used for storage? */
  idith;        /* counter for dither points */
long
  pindithsum = 0,       /* accumulator for dither average */
  pindith_a[30];        /* storage array for dither points to be averaged */
```

```
unsigned long
  ligc = 0,     /* indicates at which point to change gain */
  bsmark = 0,   /* indicates at which point to store a value in buffer */
  ngb = 0,              /* point of last gate event */
  nin = 0;              /* total number of points acquired */
float
  ***tfeparm,           /* transfer function parmeters */
  gsvbuff[GBUFSIZ],     /* gated save buffer */
  mogain[4] = {0.01,0.1,1.0,10.0};      /* selectable monitor output gains */
double
  dT,                   /* time between samples (sec) */
  fthmin,fthmax,        /* threshold values */
  zsv[4],               /* temporary space for gated buffer values */
  fpout2,               /* output value for DAC1 before conversion to int */
/* previous intermediate output values: */
  fpiprev = 0,
  eAprev[NELEMMAX],
  eBprev[NELEMMAX],
  eTOprev[NELEMMAX],
  modfact[NTGSTEP+1],   /* temporary gain factors */
  velsum = 0.,
  zsum = 0.,
  **tfA,                /* digital filter constants for TF A */
  **tfB,                /* digital filter constants for TF B */
  **tfTO;               /* digital filter constants for turnon-on TF */
char
  /* global dislay strings: */
  tfenm[NTF][NELEMMAX][9],              /* element names */
  enm[4][3] = {"HP","LP","CU","CD"},    /* element type names */
  esnm[2][9] = {"Disabled","Enabled"};  /* status */


void intr_install(unsigned char irq);
void intr_remove(unsigned char irq);
void far interrupt (*oldhandler)();
void far interrupt handler();
void bye(int err);              /* exit routine */
int fgetline(FILE *f_fp, char line[], int *l, char tc);
double gainchange(int itg);
int DecodeTFElement(char lstr[]);
void SetTF(int itf, int id);
void SetElement(int id, int elem, int pos, float *p);
void DefHP(double *a, float *p);
void DefLP(double *a, float *p);
void DefCD(double *a, float *p);
void DefCU(double *a, float *p);
void FreeArrays(void);
void DisplayTF(FILE *fp, int tfi);


int main(void)
{
  unsigned int
    status;
  double
    modconst=MODFACT,   /* parameter change factor */
    tempfact = 1.0;     /* temporary gain modification factor */
  float
    thmin=THMIN,        /* threshold where gain is automatically raised */
    thmax=THMAX;        /* threshold where gain is automatically lowered */
  int
    ver,                /* input file version number */
    dev,                /* NI device number */
    samprate=SAMPRATE,  /* sample rate (samples/sec) */
    inputgain = 1,      /* input gain */
    extcont=EXTCONT,    /* acquisition triggered externally? */
    montf = 0,          /* TF index of channel being monitored */
    vartf = 0,          /* TF to vary */
    varps = 0,          /* element position to vary */
    varpm = 0,          /* parmeter to vary */
    aiovrflw_b,         /* was there an AI overflow error? */
```

```
    aiovrrun_b,         /* was there an AI overrun error? */
    daqup_b,            /* is analog output ready for new value */
    doautogain = FALSE, /* automatically change gain? */
    doautostart = FALSE; /* start with low gain and then automatically increase? */
  unsigned long
    sampint;            /* number of clock cycles between samples */
  char
    /* display strings: */
    chanstr[3][11] = {"B","A","Vel=0"},
    tfstr[NTF][4] = {"TF1","TF2","TF3","TF4"},  /* transfer functions */
    epnm[4][3][4] = {{"Ghf","Fl","Fu"},{"Gdc","Fl","Fu"},{"Gdc","Fl","Fu"},
                    {"Gdc","Fl","Fu"}};        /* parameter names */
  FILE
    *inf,               /* input file pointer */
    *outf;              /* output file pointer */

  int i,ie,ip,j,k,l,ln,bool,itf;
  float f1;
  long li;
  unsigned long ul;
  double d1;
  char c = 0,s1[80],lnstr[80];

  /* declare arrays */
  tfes = imatrix(0,(int)NTF-1,0,(int)NELEMMAX-1);
  tfe = imatrix(0,(int)NTF-1,0,(int)NELEMMAX-1);
  tfepos = imatrix(0,(int)NTF-1,0,(int)NELEMMAX-1);
  tfpose = imatrix(0,(int)NTF-1,0,(int)NELEMMAX-1);
  tfeparm = f3tensor(0,(int)NTF-1,0,(int)NELEMMAX-1,0,2);
  tfA = dmatrix(0,(int)NELEMMAX-1,0,2);
  tfB = dmatrix(0,(int)NELEMMAX-1,0,2);
  tfTO = dmatrix(0,(int)NELEMMAX-1,0,2);

  /* clear previous output values */
  for(i=0; i<NELEMMAX; i++) {
    eAprev[i] = 0.0;
    eBprev[i] = 0.0;
    eTOprev[i] = 0.0;
  }
  for (idith=0; idith<ndithavg; idith++)
    pindith_a[idith] = 0;
  /* define temporary gain factors */
  for(i=0; i<=NTGSTEP; i++)
    modfact[i] = pow(modconst,(double)-i);
  /* clear storage buffers */
  for(i=0; i<NSBUF; i++) {
    donebuff[i] = FALSE;
    for(l=0; l<SBUFSIZ; l++)
      savebuff[i][l] = 0;
  }
  for(i=0; i<GBUFSIZ; i++) {
    gsvbuff[i] = 0.0;
  }

  clrscr();
  printf("Reading input file.\n");
  /* open input file */
  if ((inf=fopen(INFILE,"rt")) == NULL) {
    printf("digfil - error opening input file ’%s’\n",INFILE);
    FreeArrays();
    exit(1);
  }
  /* check input file version number */
  fscanf(inf,"%i\n",&ver);
  if (ver != INFVER) {
    printf("digfil - input file has incorrect version number.\n");
    FreeArrays();
    fclose(inf);
    exit(1);
  }
  /* initialize transfer function storage space */
```

```
  for(i=0; i<NTF; i++)
    tfnelem[i] = 0;
  /* read input file */
  ln = 2; l = 80; fgetline(inf,lnstr,&l,’\n’);
  while (l > 0) {
    for(i=0; i<l && isspace(lnstr[i]) && lnstr[i]!=’\’’; i++);
        /* skip leading white space but stop at comment character */
    if (lnstr[i]==’\’’) goto NextLine;
    if (strnicmp(lnstr+i,"SampleRate",10) == 0) {
      for(i+=10; i<l && !isdigit(lnstr[i]); i++);  /* find first digit */
      for(j=i; i<l && isdigit(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      samprate = atoi(s1);
      if (samprate < 7) {
        samprate = (int)SAMPRATE;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Bad sample rate, sample rate set to default: %i samples/sec\n",samprate);
      }
    } else if (strnicmp(lnstr+i,"ExtTrig",7) == 0) {
      for(i+=7; i<l && !isdigit(lnstr[i]); i++);        /* find first digit */
      extcont = lnstr[i] - ’0’;
    } else if (strnicmp(lnstr+i,"NumberofAverages",16) == 0) {
      for(i+=16; i<l && !isdigit(lnstr[i]); i++);       /* find first digit */
      for(j=i; i<l && isdigit(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      ndithavg = atoi(s1);
      if (ndithavg < 1 || ndithavg > 30) {
        ndithavg = NDITHAVG;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Number of Averages out of range, set to default: %i\n",ndithavg);
      }
    } else if (strnicmp(lnstr+i,"ParameterChangeFactor",21) == 0) {
      for(i+=21; i<l && !isnumb(lnstr[i],21); i++);     /* find first number */
      for(j=i; i<l && isnumb(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      modconst = atof(s1);
      if (modconst < 1.0) {
        modconst = MODFACT;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Parameter change factor must be >= 1, set to default: %f\n",modconst);
      }
    } else if (strnicmp(lnstr+i,"MinimumThreshold",16) == 0) {
      for(i+=16; i<l && !isnumb(lnstr[i],16); i++);     /* find first number */
      for(j=i; i<l && isnumb(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      thmin = (float)atof(s1);
      if (thmin < 0.0 || thmin >= 1.0) {
        thmin = THMIN;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Minimum threshold out of range, set to default: %f\n",thmin);
      }
    } else if (strnicmp(lnstr+i,"MaximumThreshold",16) == 0) {
      for(i+=16; i<l && !isnumb(lnstr[i],16); i++);     /* find first number */
      for(j=i; i<l && isnumb(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      thmax = (float)atof(s1);
      if (thmax < thmin || thmax > 1.0) {
        thmax = THMAX;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Maximum threshold out of range, set to default: %f\n",thmax);
      }
    } else if (strnicmp(lnstr+i,"TurnOnThreshold",15) == 0) {
      for(i+=15; i<l && !isnumb(lnstr[i]); i++);        /* find first digit */
      for(j=i; i<l && isnumb(lnstr[i]); i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
```

```
      onthresh = atoi(s1);
      if (onthresh < 0) {
        onthresh = ONTHRESH;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Turn-on threshold out of range, set to default: %i\n",onthresh);
      }
    } else if (strnicmp(lnstr+i,"ATF",3) == 0) {
      for(i+=3; i<l && !isdigit(lnstr[i]); i++);        /* find first digit */
      Atf = lnstr[i] - ’1’;
      if (Atf<0 || Atf>3) {
        Atf = 0;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function identifier, set to %i.\n",Atf+1);
      }
    } else if (strnicmp(lnstr+i,"BTF",3) == 0) {
      for(i+=3; i<l && !isdigit(lnstr[i]); i++);        /* find first digit */
      Btf = lnstr[i] - ’1’;
      if (Btf<0 || Btf>3) {
        Btf = 1;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function identifier, set to %i.\n",Btf+1);
      }
    } else if (strnicmp(lnstr+i,"TurnOnTF",8) == 0) {
      for(i+=8; i<l && !isdigit(lnstr[i]); i++);        /* find first digit */
      TOtf = lnstr[i] - ’1’;
      if (TOtf<0 || TOtf>3) {
        TOtf = 3;
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function identifier, set to %i.\n",TOtf+1);
      }
    } else if (strnicmp(lnstr+i,"tf",2) == 0) {
      /* define a new transfer function element */
      for(i+=2; i<l && !isdigit(lnstr[i]); i++);        /* find first digit */
      itf = lnstr[i] - ’1’;
      if (itf<0 || itf>3) {
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function identifier, element definition ignored!\n");
        goto NextLine;
      }
      if (tfnelem[itf] >= NELEMMAX) {
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  No more than %i elements per transfer function",NELEMMAX);
    printf("allowed, definition ignored!\n");
        goto NextLine;
      }
      /* read in element name */
      for(i++; i<l && lnstr[i]!=’"’; i++);              /* find ’"’ */
      for(j=++i; i<l && lnstr[i]!=’"’; i++)
        tfenm[itf][tfnelem[itf]][i-j] = lnstr[i];
      tfenm[itf][tfnelem[itf]][i-j] = ’\0’;
      /* read in element status */
      for(i++; i<l && isspace(lnstr[i]); i++);  /* skip white space */
      switch(toupper(lnstr[i])) {
       case ’E’: tfes[itf][tfnelem[itf]] = TRUE; break;
       default:
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function element status, set to DISABLED.\n");
       case ’D’: tfes[itf][tfnelem[itf]] = FALSE;
      }
      /* read in element type */
      for(j=++i; i<l && lnstr[i]!=’(’; i++)
        s1[i-j] = lnstr[i];
      s1[i-j] = ’\0’;
      tfe[itf][tfnelem[itf]] = DecodeTFElement(s1);
      if (tfe[itf][tfnelem[itf]] == -1) {
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Unknown transfer function element, element definition ignored!\n");
        goto NextLine;
      }
      /* read in element parameters */
      k = 0;
```

```
      do {
        for(j=++i; i<l && lnstr[i]!=',' && lnstr[i]!=')'; i++)
          s1[i-j] = lnstr[i];
        s1[i-j] = '\0';
        tfeparm[itf][tfnelem[itf]][k] = (float)atof(s1);
        if (tfeparm[itf][tfnelem[itf]][k] < 0.0) {
      printf("digfil WARNING (input file line %i):\n",ln);
      printf("  Illegal transfer function elelement paramete, ");
      printf("element definition ignored!\n");
          goto NextLine;
        }
        k++;
      } while(lnstr[i] != ')' && i < l);
      if (k != nparme[tfe[itf][tfnelem[itf]]]) {
    printf("digfil WARNING (input file line %i):\n",ln);
    printf("  Incorrect number of transfer function parameters, ");
    printf("element definition ignored!\n");
        goto NextLine;
      }
      tfpose[itf][tfnelem[itf]] = tfnelem[itf];
      tfnelem[itf]++;
    } else {
      if (i != 1)
        printf("digfil WARNING (input file line %i): unknown command\n",ln);
    }
NextLine:
    ln++; l = 80; fgetline(inf,lnstr,&l,'\n');
  }
  fclose(inf);
/*
  tfepos[itf][ie] is an array which defines the position of element
number "ie" in transfer function "itf".  Initially, this array is set
to match the order in which the elements are defined in the input file.
However, the operator may move the elements around, so invert this array
to make another array, tfpose[itf][ipos], which gives the index number of
the element to be located at position "ipos". */
  for(itf=0; itf<NTF; itf++)
    for(ie=0; ie<tfnelem[itf]; ie++)
      tfepos[itf][tfpose[itf][ie]] = ie;

  dithfact = (ndithavg < 2) ? 1 : ndithavg;
  fthmin = (double)thmin * MAXVAL;
  fthmax = (double)thmax * MAXVAL;
  dT = (double)1.0/samprate*ndithavg;
  SetTF(Atf,bab);
  SetTF(Btf,!bab);
  SetTF(TOtf,2);

  printf("Looking for the National Instruments AT-MIO-16XE-50 board...");
  dev = 0;
  bool = TRUE;
  do {
    dev++;
    if (Get_DAQ_Device_Info(dev,ND_DEVICE_TYPE_CODE,&ul))
      bool = ((int)ul != AT_MIO_16XE_50);
  } while (bool && (dev < 16));
  if (bool) {
    printf("NOT found!\n");
    FreeArrays();
    exit(1);
  } else
    printf("found.\n");

  if (irqmap[NIintr] == -1) {
    printf("Interupt IRQ%i is not supported by the AT-MIO-16XE-50!\n",(int)NIintr);
    exit(1);
  }
/* The NI-DAQ routine Init_DA_Brds() and require this call: */
  errchk(USE_E_Series_Misc(),"USE_E_Series_Misc");
/* Init_DA_Brds(deviceNumber,&deviceNumberCode) */
  errchk(Init_DA_Brds(dev,&i),"Init_DA_Brds");
```

```
  printf("NI board reset.\n");
/*
  printf("Using calibration constants from: ");
/*  printf("Factory Default.\n");
  errchk(MIO_Calibrate(dev,1,0,6,0,0,1,(double)10.0,6),"MIO_Calibrate");
   /* use factory default calibration settings */
/*
  printf("User Area #1.\n");
  errchk(MIO_Calibrate(dev,1,0,1,0,0,1,(double)10.0,6),"MIO_Calibrate");
    /* use calibration settings in User Area 1 */

  intr_install(NIintr);
  printf("Interrupt handler installed.\n");

/*
** Configure the AT-MIO-16XE-50 board **
*/
  if (printlevel > 10) printf("Write strobes 0, 1...\n");
  outwin(waWrtStrb0,0x0001);    /* strobe Write_Stobe_0 */
  outwin(waWrtStrb1,0x0001);    /* strobe Write_Stobe_1 */
/* Configure analog intput scan sequence: */
  if (printlevel > 10) printf("Load in scan sequence...\n");
  outwin(waConfMemClear,0x0000);        /* clear AI configuration memory */
  outport(aCFGMEMHigh,0x1000);
    /* Inputs are differential, bank 1 disabled, bank 0 enabled, inputs
        come from ACh0 and Ach8 */
  outport(aCFGMEMLow,0x8201);
    /* Last channel, do not generate trigger, dither enabled (required for
        AT-MIO-16XE-50), bipolar, gain = 1 */
  inputgain = 1;
/* Preconfigure clock: 100KHz internal timebase */
  if (printlevel > 10) printf("MSC Clock Configure...\n");
  ClkFOUT |= 0x0800;            /* set Slow_Internal_Timebase */
  ClkFOUT |= 0x1000;            /* set Slow_Internal_Time_Divide_By_2 */
  ClkFOUT |= 0x0100;            /* set Clock_to_Board */
  ClkFOUT |= 0x0200;            /* set Clock_to_Board_Divide_by_2 */
  outwin(waClkFOUT,ClkFOUT);
/* Reset and Initialize the DAQ-STC: */
  BoardInitialize(printlevel);
/* Configure interrupts and gating: */
  if (printlevel > 10) printf("Configure interrupts and gating...\n");
  AIMode3 &= 0xFF3F;            /* AI_FIFO_Mode = 0: interrupt on FIFO not empty */
  AIMode3 &= 0xFFE0;            /* AI_External_Gate_Select = 0: (External gating disabled) */
  outwin(waAIMode3,AIMode3);
/* Configure trigger: */
  if (printlevel > 10) printf("Configure trigger...\n");
  AIMode1 |= 0x0001;            /* set AI_Trigger_Once: not re-triggerable */
  outwin(waAIMode1,AIMode1);
  if (extcont) {
    AITrigSel &= 0xFFE0;
    AITrigSel |= 0x0001;  /* AI_START1 = 1: trigger from PFI0/TRIG1 */
    AITrigSel &= 0xF07F;
    AITrigSel |= 0x0080;  /* AI_START2 = 1: trigger from PFI0/TRIG1 */
  } else {
    AITrigSel &= 0xFFE0;  /* AI_START1 = 0: trigger from AI_START1_Pulse */
    AITrigSel &= 0xF07F;  /* AI_START2 = 0: trigger from AI_START1_Pulse */
  }
  AITrigSel &= 0x7FFF;  /* clear AI_START1_Polarity: active on positive */
  AITrigSel |= 0x0020;  /* set AI_START1_Edge: edge-sensitive */
  AITrigSel |= 0x0040;  /* set AI_START1_Sync: synchronize to SC source */
  AITrigSel &= 0xBFFF;  /* clear AI_START2_Polarity: active on positive */
  AITrigSel |= 0x1000;  /* set AI_START2_Edge: edge-sensitive */
  AITrigSel |= 0x2000;  /* set AI_START2_Sync: synchronize to SC source */
  outwin(waAITrigSel,AITrigSel);
/* Set number of scans: (continuous) */
  if (printlevel > 10) printf("Set number of scans...\n");
  AIMode1 |= 0x0002;            /* set AI_Continuous: continuous operation */
  outwin(waAIMode1,AIMode1);
  AIMode2 &= 0xDFFF;            /* clear AI_Pre_Trigger */
  AIMode2 &= 0xFFFB;            /* clear AI_SC_Intial_Load_Source: load from register A */
  AIMode2 &= 0xFFFD;            /* clear AI_SC_Reload_Mode: do not switch SC load register on TC */
```

```
  outwin(waAIMode2,AIMode2);
  outwin(waAISCLoadA,0x0000);   /* AI_SC_Load_A = 0: */
  outwin(waAISCLoadA+1,0x0000); /*  # of posttrigger scans - 1 */
  outwin(waAIComm1,0x0020);     /* strobe AI_SC_Load */
/* set start of scan conditions: internal, sample rate */
  if (printlevel > 10) printf("Set scan start condition...\n");
  AIMode3 &= 0xEFFF;    /* clear AI_SI_Special_Trigger_Delay: do not block START pulses */
  outwin(waAIMode3,AIMode3);
  AISSSel &= 0xFFE0;    /* AI_START_Select = 0: use SI_TC */
  AISSSel |= 0x0020;    /* set AI_START_Edge: edge sensitive */
  AISSSel |= 0x0040;    /* set AI_START_Sync: synchronize START */
  AISSSel &= 0x7FFF;    /* clear AI_START_Polarity: active on positive */
  outwin(waAISSSel,AISSSel);
  AIMode1 &= 0xF83F;    /* AI_SI_Source_Select = 0: AI_IN_TIMEBASE */
  AIMode1 &= 0xFFEF;    /* clear AI_SI_Source_Polarity: rising edge */
  outwin(waAIMode1,AIMode1);
  outwin(waAISILoadA,0x0000);
  outwin(waAISILoadA+1,0x0000);
  outwin(waAIComm1,0x0200);             /* strobe AI_SI_Load */
  sampint = (unsigned long)floor(20000000 / samprate);          /* 20 MHz clock */
  outwin(waAISILoadA,(int)((sampint-1)/65536));
  outwin(waAISILoadA+1,(int)((sampint-1)%65536));
  outwin(waAIComm1,0x0200);             /* strobe AI_SI_Load */
  AIMode2 &= 0xFF7F;    /* clear AI_SI_Load_Source: use register A */
  AIMode2 &= 0xFF8F;    /* AI_SI_Reload_Mode = 0: no change of SI register */
  outwin(waAIMode2,AIMode2);
  AIMode2 &= 0xFFF7;    /* clear AI_SI_Write_Switch */
  outwin(waAIMode2,AIMode2);
/* set end of scan conditions: */
  if (printlevel > 10) printf("Set scan end conditions...\n");
  AISSSel &= 0xF07F;    /* AI_Stop_Select = 31: logic low */
  AISSSel |= 0x0F80;
  AISSSel &= 0xDFFF;    /* clear AI_STOP_Sync: don't synchronize */
  AISSSel &= 0xEFFF;    /* clear AI_STOP_Edge: level-sensitive */
  AISSSel |= 0x4000;    /* set AI_STOP_Polarity: active on positive */
  outwin(waAISSSel,AISSSel);
/* set convert conditions: */
  if (printlevel > 10) printf("Set convert conditions...\n");
  outwin(waJointReset,0x0010);  /* AI_Configuration_Start */
  AIMode2 &= 0x7FFF;    /* clear AI_SC_Gate_Enable */
  AIMode2 &= 0xBFFF;    /* clear AI_Start_Stop_Gate_Enable */
  AIMode3 &= 0xF7FF;    /* clear AI_SI2_Source_Select: use same as SI */
  outwin(waAIMode3,AIMode3);
  outwin(waAISI2LoadA,0x0001);  /* AI_SI2_Load_A = 2 */
  AIMode2 &= 0xFDFF;    /* clear AI_SI2_Intial_Load_Source: use reg. A */
  outwin(waAISI2LoadB,0x0001);  /* AI_SI2_Load_B = 2 */
  AIMode2 |= 0x0100;    /* set AI_SI2_Reload_Mode: alternate first period on every stop */
  outwin(waAIMode2,AIMode2);
  outwin(waAIComm1,0x0800);     /* strobe AI_SI2_Load */
  AIMode2 |= 0x0200;    /* set AI_SI2_Intial_Load_Source: use reg. B */
  outwin(waAIMode2,AIMode2);
  outwin(waJointReset,0x0100);  /* AI_Configuration_End */
/* enable interrupts: */
  if (printlevel > 10) printf("Program and enable interrupts...\n");
  IntAEnab |= 0x0080;   /* set AI_FIFO_Interrupt_Enable */
  IntAEnab &= 0xFFF7;   /* clear AI_START_Interrupt_Enable */
  IntAEnab &= 0xFFEF;   /* clear AI_STOP_Interrupt_Enable */
  IntAEnab &= 0xFFFE;   /* clear AI_SC_TC_Interrupt_Enable */
  IntAEnab &= 0xFFFD;   /* clear AI_START1_Interrupt_Enable */
  IntAEnab &= 0xFFFB;   /* clear AI_START2_Interrupt_Enable */
  IntAEnab &= 0xFFBF;   /* clear AI_Error_Interrupt_Enable */
  outwin(waIntAEnab,IntAEnab);
/* program Interrupt Control Module: */
  IntCont &= 0xF8FF;    /* Interrupt_A_Output_Select = irqmap[NIintr] */
  IntCont |= irqmap[NIintr]*256;  /*  use interrupt channel NIintr */
  IntCont |= 0x0800;    /* set Interrupt_A_Enable */
  outwin(waIntCont,IntCont);
/* arm analog input counters: */
  if (printlevel > 10) printf("Arm analog input counters...\n");
  outwin(waAIComm1,0x0400);     /* strobe AI_SI_Arm */
  outwin(waAIComm1,0x1000);     /* strobe AI_SI2_Arm */
```

```
/* configure analog outputs: */
  if (printlevel > 10) printf("Configure analog outputs...\n");
  outport(aAOConf,0x0001);      /* DAC0: bipolar */
  outport(aAOConf,0x0101);      /* DAC1: bipolar */
  outport(aDAC0,0);
  outport(aDAC1,0);

  if (printlevel > 10) printf("Clear input and output FIFO buffers...\n");
  outwin(waADCClear,0x0000);    /* clear input FIFO buffer */
  outwin(waDACClear,0x0000);    /* clear output FIFO buffer */

  printf("NI board sampling at %i points/sec",samprate);
  if (extcont) printf(" (EXTERNALLY driven)\n");
  else printf(" (INTERNALLY driven)\n");
  if (ndithavg > 1)
    printf("Post-sample averaging the previous %i samples.\n",ndithavg);
  else
    printf("No post-sample averaging.\n");
  gotoxy(1,25);
  printf("Press any key to continue...");
  getch();
  clrscr();
  printf("  Transfer function for channel A:\n");
  DisplayTF(stdout,Atf);
  gotoxy(1,25);
  printf("Press any key to continue...");
  getch();
  clrscr();
  printf("  Transfer function for channel B:\n");
  DisplayTF(stdout,Btf);
  gotoxy(1,25);
  printf("Press any key to continue...");
  getch();
  clrscr();
  printf("  Turn-On transfer function:\n");
  DisplayTF(stdout,TOtf);
  gotoxy(1,25);
  printf("Press any key to continue...");
  getch();

  clrscr();
  printf("Status registers before starting:\n");
  ShowStatus();
  printf("Press any key to start acquistion and then 'q' to stop.\n");
  getch();
  printf("Running...\n");
  outwin(waAIComm2,0x0001);     /* strobe AI_START1_Pulse */
  printf("Status registers after starting:\n");
  ShowStatus();

  do {
    do {
      if (cstate == 3) {
        cstate = 1;
        nin = 0;
        printf("  CLOSED (compensation: channel %s using %s).\n",chanstr[bab],
               tfstr[(bab)?Atf:Btf]);
      } else if (cstate == 5) {
        cstate = 1;
        printf("  DONE (new compesation: channel %s using %s).\n",chanstr[bab],
               tfstr[(bab)?Atf:Btf]);
      } else if (cstate == 7) {
        cstate = 1;
        printf("  active channel: %s.\n",chanstr[bab]);
      }
      status = inwin(waAIStat1);
      aiovrrun_b = status & 0x0800;
      aiovrflw_b = status & 0x0400;
      if ((doautogain || doautostart) && (nin >= ligc)) {
        if ((israiled == UNDERRAIL) && (itempgain < 0)) {
          tempfact = gainchange(++itempgain);
```

```
          if (doautostart && (itempgain == 0)) {
            doautostart = FALSE;
            printf("Final total gain reached.  %s compensation active.\n",tfstr[(bab)?Atf:Btf]);
          } else {
            ligc = nin + 5*samprate;
            printf("Automatically increasing total gain (GAIN * %6.4lf) at point %li.\n",\
                   tempfact,nin);
          }
        } else if (israiled == OVERRAIL) {
          ligc = nin + samprate;
          tempfact = gainchange(--itempgain);
          printf("Automatically DECREASING total gain (GAIN * %6.4lf) at point %li.\n",
                 tempfact,nin);
        }
      }
/* Automatically turn on gain control after a certain amount of time running */
      if ((buffsel >= 0) && (nin >= bsmark)) {
        savebuff[buffsel][isbuff[buffsel]++] = (int)(fpout2*mogain[imog]);
/*      printf("Storing into buff #%1i at point %li.\n",buffsel+1,nin); */
        if (isbuff[buffsel] >= SBUFSIZ) {
          printf("Buffer #%1i full.  Storage stopped.\n",buffsel+1);
          donebuff[buffsel] = TRUE;
        } else {
          bsmark = nin + samprate;      /* every second */
        }
      }
      if (gberr == 1) {
        gberr = 2;
        printf("Gated storage buffer full.  Storage disabled.\n");
      } else if (gberr == 3) {
        gberr = 0;
        printf("WARNING: a gate (igate = %u) ocurred more than 400 ms after the previous gate!\n",
               igate);
      }
    } while (!aiovrrun_b && !aiovrflw_b && !kbhit());
    if (aiovrrun_b || aiovrflw_b) {
      if (aiovrrun_b) printf("WARNING: analog input OVERRUN error!\n");
      if (aiovrflw_b) printf("WARNING: analog input OVERFLOW error!\n");
      ShowStatus();
      outwin(waIntAAck,0x2000); /* strobe AI_Error_Interrupt_Ack */
      bye(1);
    } else {
      c = getch();
      switch(c) {
       case('A'):       /* wait for v=0 and then close loop */
        if (cstate == 0) {
          cstate = 2;
          printf("Waiting for vel = 0 to close the feedback loop...\n");
        } else if (cstate == 2) {
          cstate = 1;
          nin = 0;
          printf("  CLOSED (compensation: channel %s using %s).\n",
                 chanstr[bab],tfstr[(bab)?Atf:Btf]);
        }
        break;
       case('a'):       /* reduce gain, close loop, slowly increase gain */
        if (cstate == 0) {
          tempfact = gainchange(-NTGSTEP);
          printf("Automatically closing feedback loop...");
          doautostart = TRUE;
          itempgain = -NTGSTEP;
          ligc = 5*samprate;
          for(li=nin; (nin-li)<5*samprate;);    /* delay 5 seconds */
          nin = 0;
          printf(" CLOSED.\n");
          printf("  Compensation: %s, initial gain %6.4lf\n",
                 tfstr[(bab)?Atf:Btf],tempfact);
          cstate = 1;
        }
        break;
       case('D'):       /* open feedback loop suddenly */
        cstate = 0;
        printf("feedback loop OPEN.\n");
        itempgain = 0;
        tempfact = gainchange(itempgain);
        doautostart = FALSE;
        break;
       case('d'):       /* slowly reduce gain, open FB loop */
        if (cstate == 1) {
          printf("Gradually disabling compensation");
          while (itempgain > -NTGSTEP) {
            li = nin;
            gainchange(--itempgain);
            putchar('.');
            while ((nin-li)<samprate);  /* delay ~1 second */
          }
          cstate = 0;
          printf(" feedback loop OPEN.\n");
          doautostart = FALSE;
          itempgain = 0;
          tempfact = gainchange(itempgain);
        }
        break;
       case('1'): case('2'): case('3'): case('4'):  /* select TF for ch. A */
       case('!'): case('@'): case('#'): case('$'):  /* select TF for ch. B */
        if (!doautostart) {
          bool = FALSE;
          switch(c) {
           case('!'): i = 0; break;
           case('@'): i = 1; break;
           case('#'): i = 2; break;
           case('$'): i = 3; break;
           default: i = c - '1'; bool = TRUE; break;
          }
          if (i != ((bool)?Atf:Btf)) {  /* different from current setting? */
            if ((bool != bab) || (cstate == 0) ||
                ((cstate==4) && (i==tfreq))) {
              /* change inactive TF, FB loop open,
                 or this is 2nd request to wait for v=0 */
              if (cstate == 4) cstate = 1;
              if (bool) {
                SetTF(i,1);
                Atf = i;
              } else {
                SetTF(i,0);
                Btf = i;
              }
              if (monab == bool) monpos = 0;
                /* if user was monitoring the channel whose TF just changed,
                   the monitor position "monpos" may no longer be valid.
                   Reset it to a safe value. */
              printf("Channel %c TF changed to %s.\n",(bool)?'A':'B',
                     tfstr[(bool)?Atf:Btf]);
            } else if (cstate == 1) {
              /* change active TF and FB closed -> wait for v=0 */
              cstate = 4;
              tfreq = i;
              printf("Channel %s TF change requested: %s --> %s.\n",
                     chanstr[bab],tfstr[(bab)?Atf:Btf],tfstr[tfreq]);
            }
          }
        }
        break;
       case('u'):       /* toggle active channel: A<->B */
        if (!doautostart) {
          if ((cstate == 0) || (cstate == 6)) {
            /* FB loop open or 2nd reqest to wait for v=0 */
            if (cstate == 6) cstate = 1;
            bab = !bab;
            printf("Active channel changed to %s running %s.\n",
                   chanstr[bab],tfstr[(bab)?Atf:Btf]);
          } else if (cstate == 1) {
            /* FB loop close -> wait for v=0 */
            cstate = 6;
            printf("Active channel change requested...\n");
          }
        }
        break;
       case('p'):       /* select channel to monitor */
        monpos = 1;
        if (++monab > 2) monab = 0;
       case('o'):       /* select element of a TF to monitor */
        if (monab == 2)
          montf = TOtf;
        else
          montf = (monab) ? Atf : Btf;
        if (c == 'o')
          if (++monpos > tfnelem[montf]) monpos = 0;
        if (monpos == 0)
          printf("Monitoring input: Gain=%i, SampleRate=%i\n",inputgain,samprate);
        else {
          ie = tfepos[montf][monpos-1];         /* element index */
          printf("Monitoring channel %s",chanstr[monab]);
          if (monab < 2) printf("(%s)",(bab==monab)?"Active":"Inactive");
          printf("; position %i of %s:\n  ",monpos,tfstr[montf]);
          printf("\"%s\" %s(%.4f",tfenm[montf][ie],enm[tfe[montf][ie]],
                 tfeparm[montf][ie][0]);
          for(i=1; i<nparme[tfe[montf][ie]]; i++)
            printf(",%.4f",tfeparm[montf][ie][i]);
          printf(") %s\n",esnm[tfes[montf][ie]]);
        }
        break;
       case('O'):       /* change gain of output monitor */
        imog++;
        if (imog > 3) imog = 0;
        printf("Monitor output (DAC1) gain:%6.2f\n",mogain[imog]);
        break;
       case('c'): case('v'): case('b'):
        if (c == 'c') {                 /* cycle TF to be modified */
          if (++vartf >= NTF) vartf = 0;
          varps = 0;
          varpm = 0;
        } else if (c == 'v') {          /* cycle element to be modified */
          if (++varps >= tfnelem[vartf]) varps = 0;
          varpm = 0;
        } else if (c == 'b') {          /* cycle parameter to be modified */
          if (++varpm >= nparme[tfe[vartf][tfepos[vartf][varps]]]+2)
            varpm = 0;
        }
        ie = tfepos[vartf][varps];              /* element index */
        printf("%s element %s",tfstr[vartf],tfenm[vartf][ie]);
        if (varpm == nparme[tfe[vartf][ie]]+1)
          printf(" in position %i\n",varps+1);
        else if (varpm == nparme[tfe[vartf][ie]])
          printf(" %s\n",esnm[tfes[vartf][ie]]);
        else
          printf(": parameter %s = %f\n",epnm[tfe[vartf][ie]][varpm],
                 tfeparm[vartf][ie][varpm]);
        break;
       case('z'): case('x'):    /* modified selected parameter */
        if (!doautostart) {
          ie = tfepos[vartf][varps];            /* element index */
          if (varpm == nparme[tfe[vartf][ie]]+1) {
          /* change element position: */
            j = varps;
            if (c == 'z') {
              if (--varps < 0) varps = tfnelem[vartf] - 1;
            } else {
              if (++varps >= tfnelem[vartf]) varps = 0;
            }
            /* If this TF is active, bring the current output of this element
               along with it: */
            if (vartf==Atf) {
              d1 = eAprev[j];
              eAprev[j] = eAprev[varps];
              eAprev[varps] = d1;
            } else if (vartf==Btf) {
              d1 = eBprev[j];
              eBprev[j] = eBprev[varps];
              eBprev[varps] = d1;
            } else if (vartf == TOtf) {
              d1 = eTOprev[j];
              eTOprev[j] = eTOprev[varps];
              eTOprev[varps] = d1;
            }
            /* update index arrays: */
            tfpose[vartf][ie] = varps;
            tfpose[vartf][tfepos[vartf][varps]] = j;
            tfepos[vartf][j] = tfepos[vartf][varps];
            tfepos[vartf][varps] = ie;
            printf("%s of %s moved from position %i to %i\n",tfenm[vartf][ie],
                   tfstr[vartf],j+1,varps+1);
          } else if (varpm == nparme[tfe[vartf][ie]]) {
          /* change element status (enabled/disabled): */
            bool = tfes[vartf][ie];
            if ((c=='z' && bool) || (c=='x' && !bool)) {
              tfes[vartf][ie] = !bool;
              printf("%s of %s: %s --> %s\n",tfenm[vartf][ie],
                     tfstr[vartf],esnm[bool],esnm[!bool]);
            }
          } else {
          /* change numerical parameter: */
            f1 = tfeparm[vartf][ie][varpm];
            if (c == 'z')
              tfeparm[vartf][ie][varpm] /= modconst;
            else
              tfeparm[vartf][ie][varpm] *= modconst;
            if (vartf==Atf)
              SetElement(1,tfe[vartf][ie],varps,tfeparm[vartf][ie]);
            else if (vartf==Btf)
              SetElement(0,tfe[vartf][ie],varps,tfeparm[vartf][ie]);
            else if (vartf==TOtf)
              SetElement(2,tfe[vartf][ie],varps,tfeparm[vartf][ie]);
            printf("%s of %s from %s: %f --> %f\n",
                   epnm[tfe[vartf][ie]][varpm],tfenm[vartf][ie],
                   tfstr[vartf],f1,tfeparm[vartf][ie][varpm]);
          }
        }
        break;
       case 'g':        /* toggle autogain feature */
        if (doautogain) {
          if (itempgain && !doautostart) {
            itempgain = 0;
            tempfact = gainchange(itempgain);
          }
          printf("Automatic gain control OFF.\n");
          doautogain = FALSE;
        } else {
AutoGain: doautogain = TRUE;
          printf("Automatic gain control ON (thresholds: %5.3f,%5.3f).\n",thmin,thmax);
        }
        break;
       case 's':        /* select storage buffer */
        buffsel++;
        if (buffsel >= NSBUF) {
          buffsel = -1;
          printf("NO storage.\n");
        } else {
          printf("Use storage buffer #%1i.\n",buffsel+1);
        }
        break;
       case 'S':        /* start/stop storage */
        if (buffsel >= 0) {
          if (bsmark > nin) {
```

```
          bsmark = 0;
          printf("Storage stopped after %i points.\n",isbuff[buffsel]);
          donebuff[buffsel] = TRUE;
        } else {
          isbuff[buffsel] = 0;
          bsmark = nin+1;
          printf("Storing into buffer #%1i...\n",buffsel+1);
        }
      }
     break;
    case ’m’:        /* start/stop gated storage */
     if (dogatebuf) {
       dogatebuf = FALSE;
       if ((igate % 4) != 0)
         printf("WARNING: gated storage stopped in the middle of a sequence!\n");
       else
         printf("Gated storage disabled.\n");
     } else {
       if (gberr == 0) {
         velsum = 0.;
         zsum = 0.;
         dogatebuf = TRUE;
         printf("Gated storage enabled.\n");
       }
     }
     break;
    case ’M’:        /* zero accumulator for gated storage */
     velsum = 0.;
     zsum = 0.;
     printf("Gated storage accumulators cleared.\n");
     break;
    case ’n’:        /* show number of points in gated storage buffer */
     printf("%u points in gated storage buffer.\n",(unsigned int)floor(igate / 4));
     break;
    case ’N’:        /* remove last entry from the gated storage buff. */
     if ((igate > 3) && !dogatebuf) {
       igate -= 4;
       printf("Removing last point from gated storage buffer: %u points remain.\n",
              (unsigned int)floor(igate / 4));
     }
     break;
    case ’t’:        /* dump status registers of AtoD board */
     ShowStatus();
     break;
   }
  }
} while ((c != ’q’) && (c != ’Q’));
BoardInitialize(printlevel);
intr_remove(NIintr);
printf("%li\n",nin);

/* open output file */
if ((outf=fopen(OUTFILE,"wt")) == NULL) {
  printf("digfil - error opening output file ’%s’.\n",OUTFILE);
  FreeArrays();
  exit(1);
}
/* write to output file */
fprintf(outf,"SampleRate\t\t%i;\n",samprate);
fprintf(outf,"NumberofAverages\t%i;\n",ndithavg);
fprintf(outf,"’\n");
fprintf(outf,"ATF\t\t%i;",Atf+1);
if (bab)
  fprintf(outf," (Active)\n");
else
  fprintf(outf,"\n");
fprintf(outf,"BTF\t\t%i;",Btf+1);
if (!bab)
  fprintf(outf," (Active)\n");
else
  fprintf(outf,"\n");
```

```
  fprintf(outf,"TurnOnTF\t%i;\n",TOtf+1);
  for(itf=0; itf<NTF; itf++) {
    fprintf(outf,"’\n");
    for(ip=0; ip<tfnelem[itf]; ip++) {
      ie = tfepos[itf][ip];
      fprintf(outf,"tf %i \"%s\"\t %c %s(%.4f",itf+1,tfenm[itf][ie],
              (tfes[itf][ie])?’E’:’D’,enm[tfe[itf][ie]],tfeparm[itf][ie][0]);
      for(i=1; i<nparme[tfe[itf][ie]]; i++)
        fprintf(outf,",%.4f",tfeparm[itf][ie][i]);
      fprintf(outf,")\n");
    }
  }
  for(i=0; i<NSBUF; i++)
    if (donebuff[i]) {
      fprintf(outf,"Buffer #%1i (%i points):\n",i+1,isbuff[i]);
      for(l=0; l<isbuff[i]; l++)
        fprintf(outf,"%i\n",savebuff[i][l]);
    }
  l = (int) floor(igate / 4);
  if (l > 0) {
    fprintf(outf,"Position in mm (%i points):\n",l);
    for(i=0; i<l; i++)
      fprintf(outf,"% .7E\n",gsvbuff[i]*dT/ndithavg/4.0/ACCELCF);
  }
  fclose(outf);
  FreeArrays();

  return 0;
}


void far interrupt handler()
{
  int
    i,j,ie,
    gate,              /* value at digital port */
    stat;              /* contents of status register */
  unsigned int
    ui;
  long
    pindith;           /* raw input value, before dither avg */
  double
    fpin=0,            /* input value, after dither averaging */
    eA[NELEMMAX],      /* output of each element of TF A */
    eB[NELEMMAX],      /* output of each element of TF B */
    fpout1,            /* output of main TF */
    eTO[NELEMMAX],     /* output of each element of turn-on TF */
    tiout;             /* output of turn-on integrator */

  disable();
    pindith = dithfact*(long)inport(aADFIFO);
    if (ndithavg > 1) {
      pindithsum += pindith;
      pindithsum -= pindith_a[--idith];
      fpin = (double)pindithsum/ndithavg / dithfact;
    } else {
      fpin = (double)pindith / dithfact;
    }
    eA[0] = tfA[0][0]*eAprev[0] + tfA[0][1]*fpin + tfA[0][2]*fpiprev;
    for(i=1; i<tfnelem[Atf]; i++) {
      ie = tfepos[Atf][i];              /* element index */
      if (tfes[Atf][ie])
        eA[i] = tfA[i][0]*eAprev[i] + tfA[i][1]*eA[i-1] + tfA[i][2]*eAprev[i-1];
      else
        eA[i] = eA[i-1];
    }
    eB[0] = tfB[0][0]*eAprev[0] + tfB[0][1]*fpin + tfB[0][2]*fpiprev;
    for(j=1; j<tfnelem[Btf]; j++) {
      ie = tfepos[Btf][j];              /* element index */
      if (tfes[Btf][ie])
        eB[j] = tfB[j][0]*eBprev[j] + tfB[j][1]*eB[j-1] + tfB[j][2]*eBprev[j-1];
```

```
      else
        eB[j] = eB[j-1];
    }
    if ((cstate >= 1) && (cstate != 2))
      fpout1 = (bab) ? eA[i-1] : eB[j-1];
    else
      fpout1 = 0.0;             /* FB loop currently open */
    israiled = UNDERRAIL;       /* default: abs(fpout1) < fthmnin */
    if (fpout1 > fthmin) {
      if (fpout1 > fthmax) {
        israiled = OVERRAIL;
        if (fpout1 > MAXVAL) fpout1 = (double)MAXVAL;
      } else {
        israiled = NORAIL;
      }
    } else if (fpout1 < (-fthmin)) {
      if (fpout1 < (-fthmax)) {
        israiled = OVERRAIL;
        if (fpout1 < MINVAL) fpout1 = (double)MINVAL;
      } else {
        israiled = NORAIL;
      }
    }
    outport(aDAC0,(int)(fpout1/16.0));
    eTO[0] = tfTO[0][0]*eTOprev[0] + tfTO[0][1]*fpin + tfTO[0][2]*fpiprev;
    for(i=1; i<tfnelem[TOtf]; i++) {
      ie = tfepos[TOtf][i];             /* element index */
      if (tfes[TOtf][ie])
        eTO[i] = tfTO[i][0]*eTOprev[i] + tfTO[i][1]*eTO[i-1] + tfTO[i][2]*eTOprev[i-1];
      else
        eTO[i] = eTO[i-1];
    }
    tiout = eTO[i-1];
    if (monpos == 0)
      fpout2 = fpin;                    /* monitoring input */
    else
      switch(monab) {
       case 0: fpout2 = eB[monpos-1]; break;
       case 1: fpout2 = eA[monpos-1]; break;
       case 2: fpout2 = eTO[monpos-1]; break;
      }
    outport(aDAC1,(int)(fpout2*mogain[imog]/16.0));
    if (cstate == 2) {
      if (fabs(tiout) <= onthresh) cstate = 3;
    } else if (cstate == 4) {
      if (fabs(tiout) <= onthresh) {
        SetTF(tfreq,bab);
        if (bab)
          Atf = tfreq;
        else
          Btf = tfreq;
        cstate = 5;
      }
    } else if (cstate == 6) {
      if (fabs(tiout) <= onthresh) {
        bab = !bab;
        cstate = 7;
      }
    }
    velsum += fpin;
    zsum += velsum;
    if (dogatebuf) {
/*      gate = BGMASK & inport(aDigIn); */
      if (gate) {
        if (gbarm == 3)
          gbarm = 2;
        else if (gbarm == 1) {
          gbarm = 0;
          ui = (unsigned int) igate % 4;
          zsv[ui] = zsum;
          if ((ui != 0) && ((nin - ngb) > 1600)) gberr = 3;     /* 400 ms */
```

```
          if (ui == 3) {
            ui = (unsigned int) floor(igate / 4);
            gsvbuff[ui] = (float) (zsv[3]-zsv[2]-zsv[1]+zsv[0]);
            if (ui >= GBUFSIZ) {
              dogatebuf = FALSE;
              gberr = 1;
            }
            velsum = 0.;
            zsum = 0.;
          }
          ngb = nin;
          igate++;
        }
      } else {
        if (gbarm == 2)
          gbarm = 1;
        else if (gbarm == 0)
          gbarm = ((igate % 4) == 0) ? 3 : 1;
      }
    }
    fpiprev = fpin;
    for(i=0; i<tfnelem[Atf]; i++)
      eAprev[i] = eA[i];
    for(i=0; i<tfnelem[Btf]; i++)
      eBprev[i] = eB[i];
    for(i=0; i<tfnelem[TOtf]; i++)
      eTOprev[i] = eTO[i];
    if (ndithavg > 1) {
      pindith_a[idith] = pindith;
      if (idith == 0) idith = ndithavg;
    }
    nin++;
/* send a no-interrupt specific End of Interrupt (EOI) to both 8259s */
  outportb(aICW_M,0x20);        /* master */
  outportb(aICW_S,0x20);        /* slave */
  enable();
}


void intr_install(unsigned char irq)
{
  int aintr;
  unsigned char c1,c2;

  if (irq < 8)
    aintr = 0x08 + irq;
  else
    aintr = 0x68 + irq;

  disable();
  oldhandler = getvect(aintr);
  setvect(aintr,handler);

  maskmaster = inportb(aMask_M);
  maskslave = inportb(aMask_S);
  if (irq < 8) {
    outportb(aMask_M,(maskmaster & ~(1<<irq)));
    outportb(aMask_S,maskslave);
  } else {
    outportb(aMask_M,maskmaster);
    outportb(aMask_S,(maskslave & ~(1<<(irq-8))));
  }
  c1 = inportb(aMask_M);
  c2 = inportb(aMask_S);
  enable();
}


void intr_remove(unsigned char irq)
{
  int aintr;
```

```
  if (irq < 8)
    aintr = 0x08 + irq;
  else
    aintr = 0x68 + irq;

  disable();
  outportb(aMask_M,maskmaster);
  outportb(aMask_S,maskslave);

  setvect(aintr,oldhandler);
  enable();
}


void bye(int err)
{
  BoardInitialize(printlevel);
  intr_remove(NIintr);
  FreeArrays();

  exit(err);
}


void DefHP(double *a, float *p)
/* digital high-pass filter using the trapezoid rule
        u[z] =   (1-f*T/2) / (1+f*T/2)  u[z-1]
               + G / (1+f*T/2)          e[z]
               - G / (1+f*T/2)          e[z-1] */
{
 double
  d1,
  f = (double)p[1],     /* corner frequency (Hz) */
  Ghf = (double)p[0];   /* high-frequency gain */

  d1 = 1.0 / (1.0+f*dT/2.0);

  a[0] = (1.0-f*dT/2.0)*d1;
  a[1] = Ghf*d1;
  a[2] = -a[1];
}


void DefLP(double *a, float *p)
/* digital low-pass filter using the trapezoid rule:
        u[z] =   (1-f*T/2) / (1+f*T/2)  u[z-1]
               + G*f*T/2 / (1+f*T/2)    e[z]
               + G*f*T/2 / (1+f*T/2)    e[z-1] */
{
 double
  d1,
  f = (double)p[1],     /* corner frequency (Hz) */
  Gdc = (double)p[0];   /* DC gain */

  d1 = 1.0 / (1.0+f*dT/2.0);

  a[0] = (1.0-f*dT/2.0)*d1;
  a[1] = Gdc*f*dT/2.0*d1;
  a[2] = a[1];
}


void DefCD(double *a, float *p)
/* digital lead compensator using the trapezoid rule
        u[z] =   (1-f1*T/2) / (1+f1*T/2)                u[z-1]
               + G*f1/f2 * (1+f2*T/2) / (1+f1*T/2)      e[z]
               - G*f1/f2 * (1-f2*T/2) / (1+f1*T/2)      e[z-1] */
{
 double
  d1, d2,
```

```
  fl = (double)p[1],    /* lower corner frequency (Hz) */
  fu = (double)p[2],    /* upper corner frequency (Hz) */
  Gdc = (double)p[0];   /* DC gain */

  d1 = 1.0/(2.0+fl*dT);
  d2 = Gdc*(fl/fu);

  a[0] = (2.0-fl*dT)*d1;
  a[1] = d2*(2.0+fu*dT)*d1;
  a[2] = -d2*(2.0-fu*dT)*d1;
}


void DefCU(double *a, float *p)
/* digital lag compensator using the trapezoid rule
        u[z] =   (1-f2*T/2) / (1+f2*T/2)                u[z-1]
               + G*f2/f1 * (1+f1*T/2) / (1+f2*T/2)      e[z]
               - G*f2/f1 * (1-f1*T/2) / (1+f2*T/2)      e[z-1] */
{
 double
  d1, d2,
  fl = (double)p[1],    /* lower corner frequency (Hz) */
  fu = (double)p[2],    /* upper corner frequency (Hz) */
  Gdc = (double)p[0];   /* DC gain */

  d1 = 1.0/(2.0+fu*dT);
  d2 = Gdc*(fu/fl);

  a[0] = (2.0-fu*dT)*d1;
  a[1] = d2*(2.0+fl*dT)*d1;
  a[2] = -d2*(2.0-fl*dT)*d1;
}


int fgetline(FILE *f_fp, char line[], int *l, char tc)
/*
  Read at most "l" characters from stream "f_fp" into string "line"
until a character "tc" is encountered.  "l" returns with the number
of characters read.
*/
{
  char c;
  int i = 0;

  if (*l == 0) return FALSE;
  while(((c=fgetc(f_fp)) != tc) && (c != EOF) && (i < *l)) {
    line[i++] = c;
  }
  if ((i==*l) && (c!=tc) && (c!=EOF)) {
    line[i-1] = '\0';
    return TRUE;
  }
  line[i] = '\0';
  *l = i;
  return FALSE;
}


double gainchange(int itg)
/*
  Change the active gain of the last enabled element of the active channel by
a factor "modconst"^"itg".  Note that this change does not affect the
definition of that element.  It only changes the coefficients in the
real-time evaluation.
*/
{
 int i,ie,itf;
 float p[3];

  itg = -itg;
  itf = (bab) ? Atf: Btf;
```

```
  /* find the last enabled element in the active TF */
  for(i=tfnelem[itf]-1; i>=0 && !tfes[itf][tfepos[itf][i]]; i--);
  ie = tfepos[itf][i];          /* element index */
  p[0] = tfeparm[itf][ie][0] * modfact[itg];
  p[1] = tfeparm[itf][ie][1];
  p[2] = tfeparm[itf][ie][2];
  SetElement(bab,tfe[itf][ie],i,p);
  return modfact[itg];
}


int DecodeTFElement(char lstr[])
/*
  Scan string "lstr" for first recognizable transfer function element.
Return an integer corresponding to the element type found:
 -1 = unknown element
  0 = HP
  1 = LP
  2 = CU
  3 = CD
The scan is case sensitive but ignores leading and trailing white space.
*/
{
  int tfe = -1;
  int i,l;

  l = strlen(lstr);
  for(i=0; i<l && isspace(lstr[i]); i++);       /* skip leading white space */
  if (i < l) {
    if (strncmp(lstr+i,"HP",2) == 0) tfe = 0;
    else if (strncmp(lstr+i,"LP",2) == 0) tfe = 1;
    else if (strncmp(lstr+i,"CU",2) == 0) tfe = 2;
    else if (strncmp(lstr+i,"CD",2) == 0) tfe = 3;
  }
  return tfe;
}


void SetTF(int itf, int id)
/*
  Convert TF parmeters indexed by "itf" into nummerical constants used
to evaluate the TF in realtime.
*/
{
 int i,ie;

  for(i=0; i<tfnelem[itf]; i++) {
    ie = tfepos[itf][i];
    SetElement(id,tfe[itf][ie],i,tfeparm[itf][ie]);
  }
}


void SetElement(int id, int elem, int pos, float *p)
/*
  Convert TF parmeters of element type "elem" found in array "p[]"
into nummerical constants used to evaluate the TF in realtime.  Store these
constants in position "pos" of the space determined by "id" as follows:
  id==0:        tfB             Active TF space A
  id==1:        tfA             Active TF space B
  id==2:        tfTO            Turn-on TF space
*/
{
  switch(id) {
   case 0:
    switch(elem) {
     case 0: DefHP(tfB[pos],p); break;
     case 1: DefLP(tfB[pos],p); break;
     case 2: DefCU(tfB[pos],p); break;
     case 3: DefCD(tfB[pos],p); break;
    }
```

```
    break;
   case 1:
    switch(elem) {
     case 0: DefHP(tfA[pos],p); break;
     case 1: DefLP(tfA[pos],p); break;
     case 2: DefCU(tfA[pos],p); break;
     case 3: DefCD(tfA[pos],p); break;
    }
    break;
   case 2:
    switch(elem) {
     case 0: DefHP(tfTO[pos],p); break;
     case 1: DefLP(tfTO[pos],p); break;
     case 2: DefCU(tfTO[pos],p); break;
     case 3: DefCD(tfTO[pos],p); break;
    }
    break;
  }
}


int isnumb(char c)
/* Is "c" a digit, decimal point, minus sign, or plus sign? */
{
  return isdigit(c) || (c=='.') || (c=='-') || (c=='+');
};


void FreeArrays(void)
{
  free_imatrix(tfes,0,(int)NTF-1,0,(int)NELEMMAX-1);
  free_imatrix(tfe,0,(int)NTF-1,0,(int)NELEMMAX-1);
  free_imatrix(tfepos,0,(int)NTF-1,0,(int)NELEMMAX-1);
  free_imatrix(tfpose,0,(int)NTF-1,0,(int)NELEMMAX-1);
  free_f3tensor(tfeparm,0,(int)NTF-1,0,(int)NELEMMAX-1,0,2);
  free_dmatrix(tfA,0,(int)NELEMMAX-1,0,2);
  free_dmatrix(tfB,0,(int)NELEMMAX-1,0,2);
  free_dmatrix(tfTO,0,(int)NELEMMAX-1,0,2);
}


void DisplayTF(FILE *fp, int tfi)
/* Display transfer function "tfi" to stream "fp" */
{
  int i, ie, ip;

  fprintf(fp,"-");
  for(ip=0; ip<tfnelem[tfi]; ip++) {
    ie = tfepos[tfi][ip];                /* element index */
    fprintf(fp,"%s-",tfenm[tfi][ie]);
  }
  fprintf(fp,"\n");
  for(ip=0; ip<tfnelem[tfi]; ip++) {
    ie = tfepos[tfi][ip];                /* element index */
    fprintf(fp,"%s: %s(%.3f",tfenm[tfi][ie],enm[tfe[tfi][ie]],
            tfeparm[tfi][ie][0]);
    for(i=1; i<nparme[tfe[tfi][ie]]; i++)
      fprintf(fp,",%.3f",tfeparm[tfi][ie][i]);
    fprintf(fp,") %s\n",esnm[tfes[tfi][ie]]);
  }
}
```

Bibliography

[1] P. Sommerfeld. *Ann. Phys.*, **51**:1, 1916.

[2] Toichiro Kinoshita. The fine structure constant. *Rep. Prog. Phys.*, **59**:1459–1492, 1996.

[3] P. J. Mohr and B. N. Taylor. Codata recommended values of the fundamental constants: 1998. *Rev. Mod. Phys.*, **72**:351–495, 2000.

[4] Jr. R. S. Van Dyck, P. B. Schwinberg, and H. G. Dehmelt. *The Electron.* Kluwer Academic, Netherlands, 1991.

[5] Scott Thomas, 2001. Private communication.

[6] Robert S. Van Dyck, Jr., Paul B. Schwinberg, and Hans G. Dehmelt. New high-precision comparison of electron and positron g factors. *Phys. Rev. Lett.*, **59**:26–29, 1987.

[7] Dean L. Farnham, Robert S. Van Dyck, Jr., and Paul B. Schwinberg. Determination of the electron's atomic mass and the proton/electron mass ratio via Penning trap mass spectroscopy. *Phys. Rev. Lett.*, **75**:3598–3601, 1995.

[8] J. L. Hall, C. J. Bordé, and K. Uehara. Direct optical resolution of the recoil effect using saturated absorption spectroscopy. *Phys. Rev. Lett.*, **37**:1339–1342, 1976.

[9] B. C. Young. *A measurement of the fine-structure constant using atom interferometry.* PhD thesis, Stanford University, 1997.

[10] Ch. J. Bordé, Ch. Salomon, S. Avrillier, A. Van Lerberghe, Ch. Bréant, D. Bassi, and G. Scoles. Optical Ramsey fringes with traveling waves. *Phys. Rev. A*, **30**:1836–1848, 1984.

[11] F. Riehle, Th. Kisters, A. Witte, J. Helmcke, and Ch. J. Bordé. Optical Ramsey spectroscopy in a rotating frame: Sagnac effect in a matter-wave interferometer. *Phys. Rev. Lett.*, **67**:177–180, 1991.

[12] Pippa Storey and Claude Cohen-Tannoudji. The Feynman path integral approach to atomic interferometry. A tutorial. *J. Phys. II France*, **4**:1999–2027, 1994.

[13] R. Friedberg and S. R. Hartmann. Billiard balls and matter-wave interferometry. *Phys. Rev. A*, **48**:1446–1472, 1993.

[14] A. Peters, K.Y. Chung, and S. Chu. Measurement of gravitational acceleration by dropping atoms. *Nature*, **400**:849–852, 1999.

[15] T. L. Gustavson, A. Landragin, and M. A. Kasevich. Rotation sensing with a dual atom-interferometer sagnac gyroscope. *Class. Quantum Grav.*, **17**:2385–2398, 2000.

[16] J. Oreg, F. T. Hioe, and J. H. Eberly. Adiabatic following in multilevel systems. *Phys. Rev. A*, **29**:690–697, 1984.

[17] U. Gaubatz, P. Rudecki, M. Becker, S. Schiemann, M. Külz, and K. Bergmann. Population switching between vibrational levels in molecular beams. *Chem. Phys. Lett.*, **149**:463–468, 1988.

[18] J. R. Kuklinski, U. Gaubatz, F. T. Hioe, and K. Bergmann. Adiabatic population transfer in a three-level system driven by delayed laser pulses. *Phys. Rev. A*, **40**:6741–6744, 1989.

[19] P. Marte, P. Zoller, and J. L. Hall. Coherent atomic mirrors and beam splitters by adiabatic passage in multilevel systems. *Phys. Rev. A*, **44**:R4118–R4121, 1991.

[20] U. Gaubatz, P. Rudecki, S. Schiemann, and K. Bergmann. Population transfer between molecular vibrational levels by stimulated Raman scattering with partially overlapping laserfields. A new concept and experimental results. *J. Chem. Phys.*, **92**:5363–5376, 1990.

[21] P. Pillet, C. Valentin, R.-L. Yuan, and J. Yu. Adiabatic population transfer in a multilevel system. *Phys. Rev. A*, **48**:845–848, 1993.

[22] John Lawall and Mara Prentiss. Demonstration of a novel atomic beam splitter. *Phys. Rev. Lett.*, **72**:993–996, 1994.

[23] Lori S. Goldner, C. Gerz, R. J. C. Spreeuw, S. L. Rolston, C. I. Westbrook, W. D. Phillips, P. Marte, and P. Zoller. Momentum transfer in laser-cooled cesium by adiabatic passage in a light field. *Phys. Rev. Lett.*, **72**:997–1000, 1994.

[24] M. Weitz, B. C. Young, and S. Chu. Atom manipulation based on delayed laser pulses in three- and four-level systems: Light shifts and transfer efficiencies. *Phys. Rev. A*, **50**:2438–2444, 1994.

[25] Martin Weitz, Brenton C. Young, and Steven Chu. Atomic interferometer based on adiabatic population transfer. *Phys. Rev. Lett.*, **73**:2563–2566, 1994.

[26] A. Peters, K.Y. Chung, and S. Chu. High precision gravity measurement using atom interferometry. *Metrol.*, **38**:25–61, 2001.

[27] D. S. Weiss. *A Precision Measurement of the Photon Recoil of an Atom using Atomic Interferometry.* PhD thesis, Stanford University, 1993.

[28] Kathryn Moler, David S. Weiss, Mark Kasevich, and Steven Chu. Theoretical analysis of velocity-selective Raman transitions. *Phys. Rev. A*, **45**:342–348, 1992.

[29] E. L. Raab, M. Prentiss, Alex Cable, Steven Chu, and D. E. Pritchard. Trapping of neutral sodium atoms with radiation pressure. *Phys. Rev. Lett.*, **59**:2631–2634, 1987.

[30] D. W. Sesko, T. G. Walker, and C. E. Weiman. Behavior of neutral atoms in a spontaneous force trap. *J. Opt. Soc. Am. B*, **8**:946–958, 1991.

[31] A. M. Steane and C. J. Foot. Laser cooling below the Doppler limit in a magneto-optical trap. *Europhys. Lett.*, **14**:231–236, 1991.

[32] A. N. Nesmeianov. *Vapor Pressure of Chemical Elements.* Elsevier, Amsterdam, 1963.

[33] N. F. Ramsey. *Molecular Beams.* Oxford Univ. Press, Oxford, 1956.

[34] J. L. Hall, L. Hollberg, T. Baer, and H. G. Robinson. Optical heterodyne saturation spectroscopy. *Appl. Phys. Lett.*, **39**:680–682, 1981.

[35] E. D. Black. An introduction to Pound-Drever-Hall laser frequency stabilization. *Am. J. Phys.*, **69**:79–87, 2001.

[36] G. White and G. M. Chin. Traveling wave electro-optic modulators. *Opt. Commun.*, **5**:374–379, 1972.

[37] W. Ertmer, R. Blatt, J. L. Hall, and M. Zhu. Laser manipulation of atomic beam velocities: Demonstration of stopped atoms and velocity reversal. *Phys. Rev. Lett.*, **54**:996–999, 1985.

[38] Y. Castin, H. Wallis, and J. Dalibard. Limit of Doppler cooling. *J. Opt. Soc. Am. B*, **6**:2046–2057, 1989.

[39] P. J. Ungar, D. S. Weiss, E. Riis, and Steven Chu. Optical molasses and multilevel atoms: Theory. *J. Opt. Soc. Am. B*, **6**:2058–2071, 1989.

[40] David S. Weiss, Erling Riis, Yaakov Shevy, P. Jeffrey Ungar, and Steven Chu. Optical molasses and multilevel atoms: Experiment. *J. Opt. Soc. Am. B*, **6**:2072–2083, 1989.

[41] J. Dalibard and C. Cohen-Tannoudji. Laser cooling below the Doppler limit by polarization gradients: Simple theoretical models. *J. Opt. Soc. Am. B*, **6**:2023–2045, 1989.

[42] Carl E. Wieman and Leo Hollberg. Using diode lasers for atomic physics. *Rev. Sci. Instrum.*, **62**:1–20, 1991.

[43] B. E. A. Saleh and M. C. Teich. *Fundamentals of Photonics.* John Wiley & Sons, New York, 1991.

[44] David Dolfi, 2000. Private communication.

[45] http://tycho.usno.navy.mil/time.html.

[46] J. M. Hensley, A. Peters, and S. Chu. Active low frequency vertical vibration isolation. *Rev. Sci. Instrum.*, pages 2735–2741, 1999.

[47] J. Vanier and C. Audoin. *The Quantum Physics of Atomic Frequency Standards.* Adam Hilger, Bristol, 1989.

[48] T. J. Sumner, J. M. Pendlebury, and K. F. Smith. Conventional magnetic shielding. *J. Phys. D*, **20**:1095–1101, 1987.

[49] David Lunt, 1999. Private communication.

[50] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, second edition, 1992.

[51] D. S. Weiss, B. C. Young, and S. Chu. Precision measurement of the photon recoil of an atom using atomic interferometry. *Phys. Rev. Lett.*, **70**:2706, 1993.

[52] D. S. Weiss, B. C. Young, and S. Chu. Precision measurement of $\hbar/m_{\mathrm{Cs}}$ based on photon recoil using laser-cooled atoms and atomic interferometry. *Appl. Phys. B*, **59**:217–256, 1994.

[53] A single repetition of the recoil measurement consists of four interferometers: $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, and $\boxed{4}$. As discussed in section 5.1, we fit the fringe data from each of these interferometers with a sinusoidal function. To calculate a value for the recoil frequency we combine the resulting four phase fit parameters and their corresponding fit uncertainties as follows. We first calculate two intermediate values:

(a) the difference between the *normal* interferometers $\boxed{1}$ (down) - $\boxed{2}$ (up) and (b) the difference between the *inverted* interferometers (all recoil directions inverted) $\boxed{3}$ (up) - $\boxed{4}$ (down), where we combine the individual uncertainties in quadrature. We then compute (c) the unweighted arithmetic mean between the normal and inverted interferometer differences: $\frac{1}{2}\left[\left(\boxed{1}-\boxed{2}\right)+\left(\boxed{3}-\boxed{4}\right)\right]$. Its uncertainty comes from the uncertainties of the normal and inverted interferemeter differences summed in quadrature and divided by two. The value (c) and its uncertainty thus represents a single repetition of the recoil frequency f_{rec}. Finally, we calculate the weighted mean of all repetitions of the *normal* interferometers (a) and plot them as solid circles ($\bullet$) without error bars. The weighted mean of all *inverted* interferometers (b) are shown as hollow circles ($\circ$), and the weighted mean of all mean values (c) are plotted with single standard deviation error bars as solid triangles (**6**). All of these values are plotted in parts per billion (ppb) relative to an arbitrary but fixed value of $f_{\mathrm{fix}} = 15\,006.278\,875$ Hz for the real value f_{rec}.

[54] A. Peters. *High precision gravity measurements using atom interferometry.* PhD thesis, Stanford University, 1998.

[55] T. Miller and B. Bederson. Atomic and molecular polarizabilities: review of recent advances. *Adv. At. Mol. Phys.*, **13**:1–55, 1977.

[56] K. J. Boller, A. Imamoglu, and S. E. Harris. Observation of electromagnetically induced transparency. *Phys. Rev. Lett.*, **66**:2593–2596, 1991.

[57] M. O. Scully and M. Fleischhauer. High-sensitivity magnetometer based on index-enhanced media. *Phys. Rev. Lett.*, **69**:1360–1363, 1992.

[58] Kurt Gibble and Steven Chu. Laser-cooled Cs frequency standard and a measurement of the frequency shift due to ultracold collisions. *Phys. Rev. Lett.*, **70**:1771–1774, 1993.

[59] S. Ghezali, P. Laurent, S. Lea, and A. Clairon. An experimental study of the spin-exchange frequency shift in a laser-cooled cesium fountain frequency standard. *Europhys. Lett.*, **36**:25–30, 1996.

[60] J. Anandan. Curvature effects in interferometry. *Phys. Rev. D*, **30**:1615–1624, 1984.

[61] J. Audretsch and C. Lammerzahl. New inertial and gravitational effects made measurable by atomic-beam interferometry. *Appl. Phys. B*, **54**:351–354, 1992.

[62] S. Wajima, M. Kasai, and T. Futamase. Post-Newtonian effects of gravity on quantum interferometry. *Phys. Rev. D*, **55**:1964–1970, 1997.

[63] T. Udem, J. Reichert, R. Holzwarth, and T. W. Haensch. Absolute optical frequency measurement of the cesium D-1 line with a mode-locked laser. *Phys. Rev. Lett.*, **82**:3568–3571, 1999.

[64] M. P. Bradley J. V. Porto, S. Rainville, J. K. Thompson, and D. E. Pritchard. Penning trap measurements of the masses Cs-133, Rb-87, Rb-85, and Na-23 with uncertainties $\leq$ 0.2 ppb. *Phys. Rev. Lett.*, **83**:4510–4513, 1999.

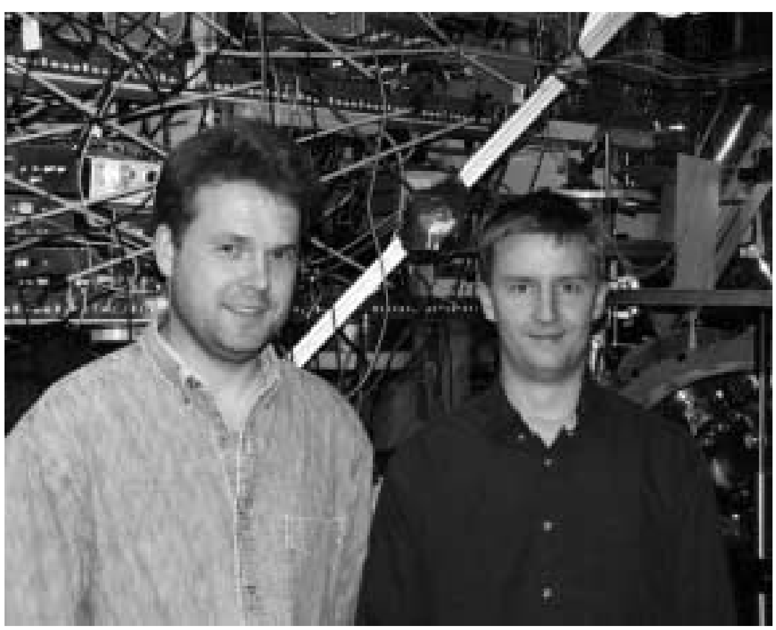
[65] J. P. Gordon and A. Ashkin. Motion of atoms in a radiation trap. *Phys. Rev. A*, **21**:1606, 1980.

[66] L. Young, W. T. Hill III, S. J. Sibener, Stephen D. Price, C. E. Tanner, C. E. Weiman, and Stephen R. Leone. Precision lifetime measurements of Cs $6p\ {}^2P_{1/2}$ and $6p\ {}^2P_{3/2}$ levels by single-photon counting. *Phys. Rev. A*, **50**:2174–2181, 1994.

[67] Bruce W. Shore. *The theory of coherent atomic excitation, Volume 2: Multilevel atoms and Incoherence.* John Wiley & Sons, New York, third edition, 1990.

Index

This photo of Steve Chu, the author, and Kurt Franke (left-to-right) was taken in Steve's office as we discussed the photon recoil measurement. It appeared in the *New York Times* Science Section. At the time, for a caption, Steve suggested, "Kurt Franke meets with two unknown physicists".



Andreas Wicht and the author in front of the photon recoil apparatus.