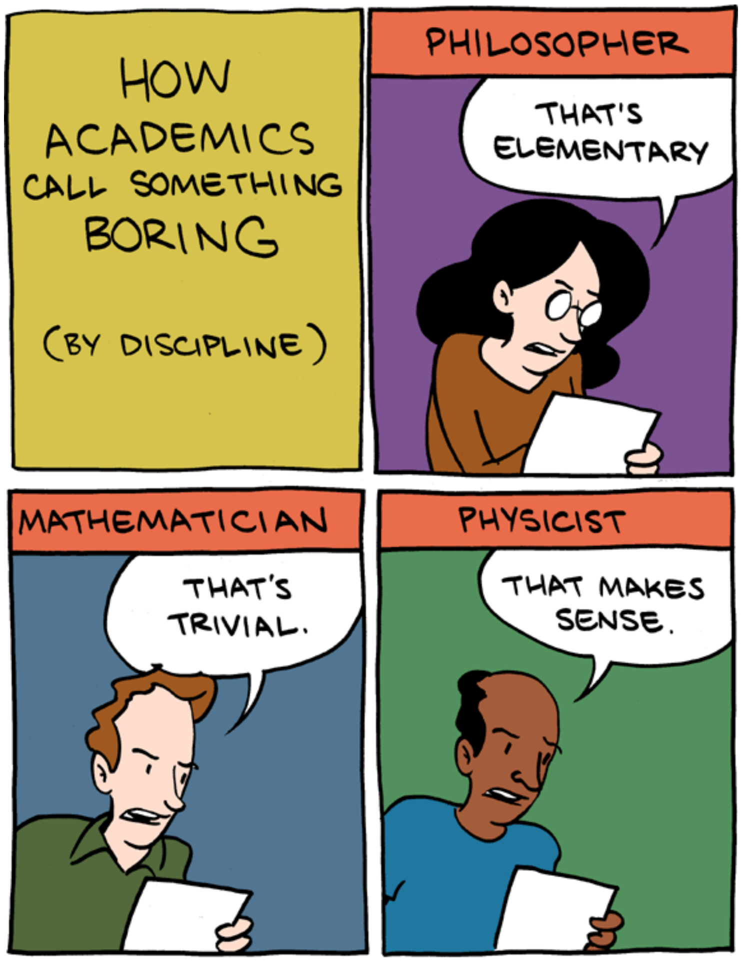
Nanophotonic Resonators for Optical Quantum Memories Based on Rare-Earth-Doped Materials

Thesis by
Evan Tsugio Miyazono

In Partial Fulfillment of the Requirements for the
degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2017
Defended April 5, 2017



Is this thesis informative and engaging? Hopefully. Is it interesting? Perhaps.
But regardless, this thesis should at least make sense.

ACKNOWLEDGEMENTS

The dynamics of any system are governed by its internal evolution and its coupling to external systems. As my state at $t = 0$ was trivial, I entirely credit the rest of the universe for this from-scratch apple pie [1], but these are the dominant terms.

Andrei, thank you for the opportunity to join your group and everything from then to now. I deeply appreciate the trust you put in me by bringing me on as your first student, and I hope I rose to the challenge. It's hard to say whether I'll more fondly remember the times when we had our first successes, or the times when it seemed clear we were both figuring out our respective roles together. I've grown in many dimensions under your guidance while deeply enjoying the experience. I've never regretted coming to Caltech and joining your group, and I think that's the best available metric for any decision.

I must express my appreciation to the whole rare earth subgroup. Thank you to Jon and Tian for knowing the answers to all nearly all of my questions about echoes and holes, and for sharing the knowledge from troubleshooting the Nd experiments. Jon, your wanderings over to 214 were more helpful than you'll ever know. Jake, thanks for taking over fab on the erbium side; you are truly a gifted nano-sculptor. John, your enthusiasm is contagious and your thesis has remained my favorite REI spectroscopy reference; I wish you could have joined us sooner. Alex, thanks for the Perl, lol. Seriously, though, I'm very fond of work we did together, both the research and the friendship that resulted from it. Ioana, I couldn't have asked for a better lab partner. Thank you for our wordless coordination, alternating which of us was tirelessly optimistic and which was exasperatedly cynical. And while I doubt it was a conscious effort, thanks for reminding me of things I value and ought to spend time on. I'm not sure how I would have made it this far without you.

Though other group members worked on unrelated topics, I can't neglect acknowledging their help. Thank you to Yu, Ehsan, and Mahsa, for the occasional assistance on KNI tools, critical advice on fabrication methods, and helping manage the simulation computers. I extend extra gratitude to Amir, for always being willing to give me a sanity check, always answering helpfully, and dispelling apparent paradoxes without me feeling judged or daft. (Well, whenever that was possible.)

I would also like to thank the members of my committee, Axel Scherer, Keith Schwab, and Kerry Vahala. I've consistently been enthralled by conversations with each of you in which you provided guidance and perspective.

I owe much gratitude to the KNI staff, and everyone from the Painter group and Scherer group who helped train me, specifically thanks to Richard, Justin, Sean, and Andrew for starting me off in the right direction with simulations and fab, and occasionally providing course corrections. Max, the instances of respite and nepenthe we stumbled across or hunted down together were no less important to this thesis than your fab training and advice you gave me.

I'd also like to thank everyone at Caltech who gave me support outside the lab, starting with Christy and Cecilia, who helped so many of us to focus on the research. Though they are too many to name, I have to thank all my Caltech friends; I promise I appreciated every single invitation you extended to me, even if I wasn't able to make it. I give many, many thanks to all my friends from what feel like previous lives. If I forced you through a lab tour, or if I flew to visit you and forced you to decipher diagrams on napkins or go through my most recent presentation slides, I promise that you helped make the explorations of an unimaginably long string of dead ends that is research become substantially less burdensome, and even exciting (especially if I photographed you in a bunny suit). And however transient it may have ended up being, I'd like to think we're all part of the same big positive feedback loop of optimism and pursuit of our respective potentials.

My family's role in shaping who I am started well before graduate school, and naturally continued throughout. Morgan, thank you for the occasional recipe idea or ingredient combination that either brightened my day or reminded me that creativity means trying things that sound absurd, sometimes simply for that reason. Clinton, I definitely appreciated the encouragement and the reminders to enjoy school while I can, as well as the various visits and trips that eased the burden during the more difficult stretches of grad school. Jeannette, that strategy you gave for writing copious amounts was more useful than you might guess. Mom, Dad, thank you for getting me here. You deserve more credit than you'll ever accept. By comparison, all other teachers and mentors have made negligible contributions to me being the person I am today, with all the accomplishments I can claim.

Kate, nothing I can say here can thank you enough for the encouragement, support, sympathy, empathy, and love you've shown me from thousands and then hundreds of miles away as I consistently "just finish the last *n* months of this Ph.D." for around 3*n* months for at least the past two years. While acknowledging that is not enough, it at least fits on the page. I'll try to make it up to you.

ABSTRACT

The growing interest in optical quantum systems has led to the exploration of multiple platforms. Though pioneering experiments were performed in trapped atom and trapped ion systems, solid state systems show promise of being scalable and robust. Rare earth dopants in crystalline hosts are an appealing option because they possess a rich spectrum of energy levels that result from a partially filled electron orbital. While level structure varies across the period, all elements possess crystal field splittings corresponding to near infra-red or optical frequencies, as well as Zeeman and often hyperfine levels separated by radio frequency and microwave frequencies. These levels demonstrate long excited-state lifetimes and coherence times and have been used in diverse applications, including demonstrating storage of a photonic state, converting of optical to microwave photons, and manipulating a single ion as a single qubit. The ions' weak interaction with their environment results in low coupling to optical fields, which had previously required measurements with macroscopically large ensembles of ions. Coupling the ions to an optical cavity enables the use of a smaller ensemble, which is required for the development of the aforementioned technologies in an on-chip scalable architecture.

This thesis contains recent progress towards fabricating optical micro and nanocavities coupled to ensembles of erbium ions, mainly erbium in yttrium orthosilicate. In one design, focused ion beam milling was used to create a triangular nanobeam photonic crystal cavity in a bulk erbium-doped substrate. A second design leveraged the fabrication capabilities of silicon photonics, defining amorphous silicon ring resonators using electron beam lithography and dry etching. These devices coupled evanescently to erbium ions below the ring, in the bulk substrate. Simulation, design, fabrication, and characterization of both resonators are discussed. Coupling between the ions and the resonator is demonstrated for each, and capabilities offered by these devices are described. Preliminary work implementing coherent control of erbium ions is presented. Lastly, alternative substrates are evaluated for possible future solid-state erbium systems.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Evan Miyazono et al. “Coupling erbium dopants in yttrium orthosilicate to silicon photonic resonators and waveguides”. In: *Optics Express* 25.3 (Feb. 2017), p. 2863. ISSN: 1094-4087. DOI: `10.1364/OE.25.002863`.
ETM participated in the conception of the project and fabricated the device. Characterization, data collection, and analysis was performed with IC. The manuscript was written by ETM with input from IC and AF.

[2] Evan Miyazono et al. “Coupling of erbium dopants to yttrium orthosilicate photonic crystal cavities for on-chip optical quantum memories”. In: *Applied Physics Letters* 108.1 (Jan. 2016), p. 011111. ISSN: 0003-6951. DOI: `10.1063/1.4939651`.
ETM participated in the conception of the project, fabricated and characterized the device, gathered and analyzed the data, and wrote the manuscript with AF.

[3] Tian Zhong et al. “High quality factor nanophotonic resonators in bulk rare-earth doped crystals”. In: *Optics Express* 24.1 (Jan. 2016), p. 536. ISSN: 1094-4087. DOI: `10.1364/OE.24.000536`.
ETM participated in simulations and some assembly of characterization optics.

[4] Tian Zhong et al. “Nanophotonic coherent light–matter interfaces based on rare-earth-doped crystals”. In: *Nature Communications* 6 (Sept. 2015), p. 8206. ISSN: 2041-1723. DOI: `10.1038/ncomms9206`.
ETM participated in simulations and assembly of characterization optics.

TABLE OF CONTENTS

Acknowledgements **iv**
Abstract **vi**
Published Content and Contributions **vii**
Table of Contents **viii**
List of Illustrations **x**
List of Tables **xii**
Nomenclature **xiii**
Chapter I: Introduction **1**
1.1 Solid State Optical Quantum Memories 1
1.2 Rare Earth Ions 4
1.3 Spectral Tailoring and Echos 13
1.4 Rare Earth Ion Cavity Quantum Electrodynamics 19
1.5 Outline of the Thesis 23
Chapter II: Triangular Nanobeam Devices **25**
2.1 Device Simulation and Design 25
2.2 Nanobeam Fabrication 28
2.3 Confocal Microscope Characterization 29
2.4 Cavity-Ion Coupling 31
Chapter III: Hybrid Microring Devices **37**
3.1 Initial Device Study for Neodymium Ions 37
3.2 Device Simulation and Design 40
3.3 Hybrid Ring Fabrication 43
3.4 Device Characterization 47
3.5 Amorphous Silicon Hydride Absorption at Shorter Wavelengths . . . 49
Chapter IV: Optical Manipulation of Erbium Ions **52**
4.1 Purcell Enhanced Decay Measurement 52
4.2 Comparison of Nanobeam and Hybrid Microring devices 58
4.3 Beyond a Two Level System 61
4.4 Photon Echoes 67
Chapter V: Erbium-167 and Future Directions **69**
5.1 Hyperfine Structure 69
5.2 Hyperfine Level Hole Burning 70
5.3 Atomic Frequency Combs 70
5.4 Summary and Future Directions 75

Bibliography **78**

Appendix A: Code for Cavity Mode Simulation and Analysis **87**

A.1 MEEP Code to Model the Ring 87

A.2 MEEP Code to Model the Nanobeam 88

A.3 Matlab Code to Analyze the Ring 91

Appendix B: Code for Running Optical Characterization Experiments **95**

B.1 Tektronix 5014 Arbitrary Waveform Generator 95

B.2 Sequence Loader 100

B.3 Tunics TECL 122

B.4 Toptica DLC pro 124

B.5 Tektronix Oscilloscope 128

LIST OF ILLUSTRATIONS

Number *Page*
1.1 Elastic and inelastic decoherence on the Bloch sphere (illustration) . . 3
1.2 Outer orbital radial electron density for a rare earth ion (theory) . . . 5
1.3 Erbium energy levels and lifting degeneracies (illustration) 8
1.4 Homogeneous and inhomogenous linewidths (illustration) 9
1.5 Yttrium orthosilicate unit cell and principal axes (illustration) 11
1.6 Spectral hole burning on the spin levels of erbium (illustration) . . . 14
1.7 Two pulse photon echo on the Bloch sphere (illustration) 16
1.8 Atomic frequency comb delay lines and spin-wave storage (illustration) 18
1.9 Cavity quantum electrodynamics rates (illustration) 19
2.1 Nanobeam cavity robustness against error (simulation) 26
2.2 Nanobeam cavity mode profile (simulation) 27
2.3 Nanobeam cavity (micrographs) 30
2.4 Confocal microscope (illustration and photo) 32
2.5 Nanobeam cavity transmission spectrum (measurement) 33
2.6 Cryostat nitrogen tuning line (photo) 34
2.7 Nanobeam cavity transmission while tuning (measurement) 35
2.8 Resonant nanobeam cavity transmission (fit) 36
3.1 Resonance of gallium arsenide rings on glass (measured) 39
3.2 Gallium arsenide device transfer (photos and micrographs) 41
3.3 Ring resonator field profile (simulation) 42
3.4 Grating coupler optimization (simulation) 43
3.5 Fabrication procedure for silicon hydride rings (illustration) 44
3.6 Amorphous silicon hydride microring (micrographs) 46
3.7 Hybrid ring resonator transmission spectra (measurement) 48
3.8 Hybrid ring transmission while tuning (measurement) 49
3.9 Resonant microring cavity transmission (fit) 50
3.10 Silicon hydride ring resonances at other wavelengths (measurement) . 51
4.1 Purcell-enhanced decay in both cavities (measurement) 54
4.2 Histograms of coupling strength for both resonators (simulation) . . . 55
4.3 Photoluminescence decay vs detuning (simulation) 56
4.4 Purcell enhancement and photoluminescence from both cavities (fit) . 57

4.5 Effective Purcell factor vs detuning (fit) 58
4.6 Coupling to devices withing the Bluefors fridge (photo) 62
4.7 Bluefors fiber access to the Bluefors fridge (photo) 63
4.8 Nanobeam scans inside the fridge (measurement) 64
4.9 Fiber optic sample characterization system (illustration) 65
4.10 Nanobeam reflection while tuning in a magnetic field (measurement) 66
4.11 Atomic frequency comb in the nanobeam cavity (measurement) . . . 67
4.12 Nanobeam echo and coherence decay curve (measurement) 68
5.1 ^{167}Er:YSO inhomogeneous line with spectral holes (measurement) . . 71
5.2 Accumulated atomic frequency combs in ^{167}Er:YSO (fit) 72
5.3 Piecewise-defined atomic frequency comb in ^{167}Er:YSO (measurement) 74
5.4 Atomic frequency comb echoes in ^{167}Er:YSO (measurement) 75

LIST OF TABLES

Number *Page*
3.1 PECVD deposition recipe for amorphous silicon hydride 44
3.2 Spin coating recipe for films for electron beam lithography 45
3.3 ICP etch recipe for amorphous silicon hydride 45
4.1 Summary of various parameters for the two optical resonator designs 59

NOMENCLATURE

β_r	the branching ratio; probability an excited state decays directly to a particular level
ϵ	the permittivity
ϵ_0	the vacuum permittivity
η	efficiency of photon memory recall
$\hbar$	the reduced Planck constant
κ_c	the rate of energy decay from the cavity into the input/output coupled mode
κ_i	the rate of energy decay from the cavity into all other modes
$\mathbf{B}$	the applied magnetic field
$\mathcal{F}$	finesse of the atomic frequency comb, defined analogously to Fabry-Pérot cavities
μ	the dipole moment of the transition in question, in the direction parallel to the polarization of the electric field
μ_B	the Bohr magneton, $\frac{e\hbar}{2m_e}$
max()	a function that returns the maximum value of the argument
Ω	ensemble coupling rate between the ions and the cavity, a generalization of g_tot
ω_c	the angular frequency of the cavity resonance
ρ	density of ions; approximately 1.87 million ions per cubic micron in 200 ppm Er:YSO
$\vec{E}(r)$	the electric field vector as a vector-valued function of position
ζ	cooperativity; figure of merit describing ensemble-cavity coupling
F_P	the Purcell factor; ratio between an emitter's rate of spontaneous emission in a cavity to the rate in free space
g_0	the coupling rate between an emitter and a cavity
g_ion	the coupling rate for an ion at the cavity electric field maximum
g_tot	total coupling rate for the ensemble

n	the refractive index
Q_i	the intrinsic quality factor, $\frac{\omega_c}{\kappa_i}$
Q_l	the loaded quality factor, $\frac{\omega_c}{\kappa_i+\kappa_c}$
V_{mode}	the mode volume of the cavity
AFC	**a**tomic **f**requency **c**omb
AOM	**a**cousto-**o**ptic **m**odulator
cQED	**c**avity **q**uantum **e**lectro**d**ynamics
EOM	**e**lectro-**o**ptic **m**odulator
HF	hydrofluoric acid; it's a chemical formula, not an acronym, silly
MEEP	**M**IT **e**lectromagnetic **e**quation **p**ropagation open source FDTD simulation software
optical excited state	the lowest energy state of the $^4I_{13/2}$ total angular momentum manifold
optical ground state	the lowest energy state of the $^4I_{15/2}$ total angular momentum manifold
PDMS	**p**oly**d**i**m**ethyl**s**iloxane
PL	**p**hoto**l**uminescence
qubit	**qu**antum **bit**
RF	**r**adio **f**requency
SEM	**s**canning **e**lectron **m**icroscope
SNSPD	**s**uperconducting **n**anowire **s**ingle **p**hoton **d**etector
telecom C band	the conventional "erbium window" in optical telecommunications from 1530-1565 nm
YSO	**y**ttrium **s**ilicon **o**xide, also yttrium orthosilicate, Y_2SiO_5

Chapter 1

INTRODUCTION

1.1 Solid State Optical Quantum Memories

The most prominent promises of engineered quantum systems are two potentially antipathetic cases of quantum information manipulation. The better popularized set of implementations are lumped under the category of quantum computation. Stemming from the idea of the classical computability of physical systems, quantum computation leverages additional complexity in quantum systems to store and process information [2, 3]. It found its first interesting engineering application in Peter Shor's algorithm for solving the discrete logarithm problem and performing integer factorization in polynomial time on a quantum computer [4]. This application won renown because an efficient implementation of this algorithm would compromise the RSA cryptosystem, which is commonly used to secure network traffic, encrypt and authenticate e-mail, and secure credit card payment systems [5].

Interestingly, the second set of imminently implementable applications of quantum engineering are quantum cryptography, which strives to provide secrecy and security by leveraging aspects of quantum mechanics. Quantum key distribution (QKD), one of the older and more developed areas of quantum cryptography, promises provably secure encryption above an arbitrarily high confidence threshold [6]. While commercial systems exist for generation and measurement of quantum bits (qubits) needed for the two main QKD algorithms [7, 8], the distance over which keys can be distributed is limited to the distance over which a quantum bit can be conveyed. For photons, this limit is dominated by absorption/distortion for fiber optics or alignment/focusing of free space optics. Due to the complexity, of free space optical communication, fiber systems are easier to construct, install over large distances, and maintain. Fortunately, the exponential decay of transmission success as a function of distance can be somewhat circumvented with a sufficiently good quantum memory. This is achieved by implementing entanglement swapping [9] and purification of entanglement to construct a quantum repeater [10], where the qubit transmission efficiency scales polynomially with distance [11].

In the process of developing optical quantum memories, various researchers conceived of additional interesting applications, including quantum metrology [12], on-demand single photons [13], efficient single-photon detection [14], and probing fundamental quantum phenomena [15].

After a brief explanation of lifetimes and coherence of quantum states, the remaining sections of this chapter will be spent on three topics. First, I will describe the physical properties that make rare earth ions a naturally superior platform for long-term storage of quantum information. Then, I will outline various methods of initializing, storing, and recovering classical and quantum information using rare earth ions. Next, I will highlight the motivations toward building a scalable interface. The chapter will conclude with an outline of the remainder of the thesis.

Lifetime and Coherence

Throughout this thesis the lifetime of quantum states and the coherence time between states will be discussed as two relevant timescales affecting the corruption of a quantum state. The lifetime of a state, denoted T_1, is the inverse of the sum of the various exponential decay rates at which that state decays to other states. Symbolically,

$$T_1 \equiv 1/\Gamma_{\text{tot}} = 1/\left(\sum \Gamma_i\right), \tag{1.1}$$

where Γ_i is one of various decay rates. These rates are related to various mechanisms relaxing the qubit to the ground state. The relaxation can involve decays via other levels in a many-level system. The exponential nature of the decay arises from the fact that the probability of decay at any moment in time is constant.

The lifetime, which describes the rate of excited state information loss, is distinct from the coherence time, which characterizes the loss of phase information usually with respect to the ground state. Before clarifying, as an analog I would like to point out that one can describe photon interference experiments in two ways. In the common explanation, which I would consider arguably incorrect, photons following one path interfere with photons passing through the other path, and the phase difference between the two photons causes the resulting interference pattern, provided that the path difference is shorter than the coherence length of the photon. This explanation breaks down if the experiment is performed with single photons, and we must conclude that the single photon is passing through both paths of the interferometer and interfering with itself. The proper explanation of multi-photon interference should similarly involve each photon interfering with itself.

I describe this because the description of coherence decay that is often used in nuclear magnetic resonance involves an ensemble of nuclear spins rotating in the

xy-plane with an applied magnetic field in the z direction. In this description, due to fluctuations and interactions, the bulk magnetization in the xy-plane becomes smeared out across the ensemble until it sums to zero. If one is dealing with a single magnetic moment or electron spin, then the xy-plane of the precessing magnetization must be described as a superposition of the excited and ground states, which possesses an associated phase between the two. Throughout this thesis, I will generally describe the decay or decoherence of the ensemble as though each ion remained a pure state; though illustrative and more conceptually intuitive, this is demonstrably not the case and explanations which break down on the single-ion level should be understood in the aforementioned manner.

Evolution of quantum two-level states is typically represented graphically as a point on the surface of a Bloch sphere, where zenith and nadir represent the ground $|g\rangle$ and excited states $|e\rangle$, respectively. The equator represents equal superposition of the two and latitudinal precession is expressed as a complex phase. In this representation, decoherence is the smearing out of the pure state latitudinally, leading to a loss of phase information. Similarly, smearing out in the vertical direction corresponds to the collapse of the state into the ground or excited state, and is thus described by T_1. An illustration of these two decay types is shown in Figure 1.1. As decay processes that destroy amplitude will also destroy the phase information, the T_1 decoherence time will also limit the T_2.

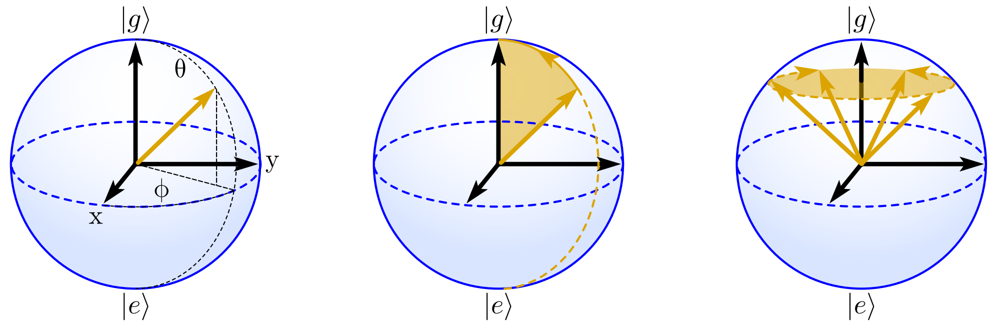


Figure 1.1: Illustration of a state vector on the Bloch sphere (left) as well as inelastic (center) and elastic (right) decoherence processes. The zenith ($\theta = 0$) and nadir of the sphere are the ground and excited states, respectively. The energy of the state is encoded in θ, while ϕ encodes the complex phase, represented as the latitude on the Bloch sphere. Thus, inelastic decoherence, characterized by T_1, is a smearing of the pure state longitudinally, while elastic decoherence involves smearing of the state latitudinally.

Using the definition of ‘elastic’ that means preserving or conserving energy, T_2

and T_1 decay are also sometimes referred to as elastic and inelastic decoherence. It's also worth mentioning explicitly that a lifetime and coherence time refer to a two level system, and though the ground state is usually understood, in some instances ambiguity arises from the presence of multiple ground states.

1.2 Rare Earth Ions

Rare earth elements, namely scandium ($_{21}$Sc), yttrium ($_{39}$Y), lanthanum ($_{57}$La), and the lanthanides ($_{57}$Ce to $_{71}$Lu) [16], are a distinctive group of elements sharing similar chemical properties. Interestingly, the misleading name "rare earth" arises from a historic difficulty to mine and purify the elements rather than any actual scarcity and a mistaken association with the alkaline earth metals. The rare earth ions share an electron configuration consisting of a Xenon electron core, a filled $6s$ orbital, a partially filled $4f$ orbital, and, for La, Ce, Gd, and Lu, a single electron in a $5d$ state; this $5d$ occupation occurs in elements that have an empty, nearly empty, half-full, and full $4f$ orbital, respectively. The chemical properties shared by the lanthanides result from a trivalent oxidation state with electron configuration [Xe]$4f^N$, where N takes values between 1 and 14 for each of the 14 lanthanides.

4f Intraorbital Transitions

Because a trivalent rare earth ion is still a multi-electron system, the partially filled $4f$ orbitals cannot merely be described using single electron wave function solutions to the hydrogen atom. Instead, the wave functions for rare earth ions in vacuum can be approximated using the Hartree-Fock method, numerically building a many-body wave function as a Slater determinant of the single particle wave functions. This has been performed for many of the rare earth ions in [17], and a more comprehensive treatment of Hartree-Fock calculations can be found therein.

Attributes of the wave functions can be calculated from the basis of energy eigenstates produced by the Hartree-Fock method. Specifically, it is interesting to examine the radial electron density distribution of the $4f$, $5s$, $5p$, and $6s$ orbitals in gadolinium, shown in Figure 1.2, and note that the partially filled $4f$ orbital are spatially located within the filled $5s$ and $5p$ orbitals. The remarkable coherence properties of the various configurations of the $4f$ states result from the $5s$ and $5p$ orbitals strongly shielding the partially filled $4f$ orbitals. These outer orbitals are often likened to a Faraday cage for electrons in the $4f$ orbitals, shielding them from external electric fields. Though this is somewhat metaphoric, the similarity between spectra for a free-ion and in a host crystal illustrates the low influence that external fields have. While Figure 1.2 specifically shows probability densities for gadolinium, the relative radial distributions vary weakly with atomic number across the rare earth ions.

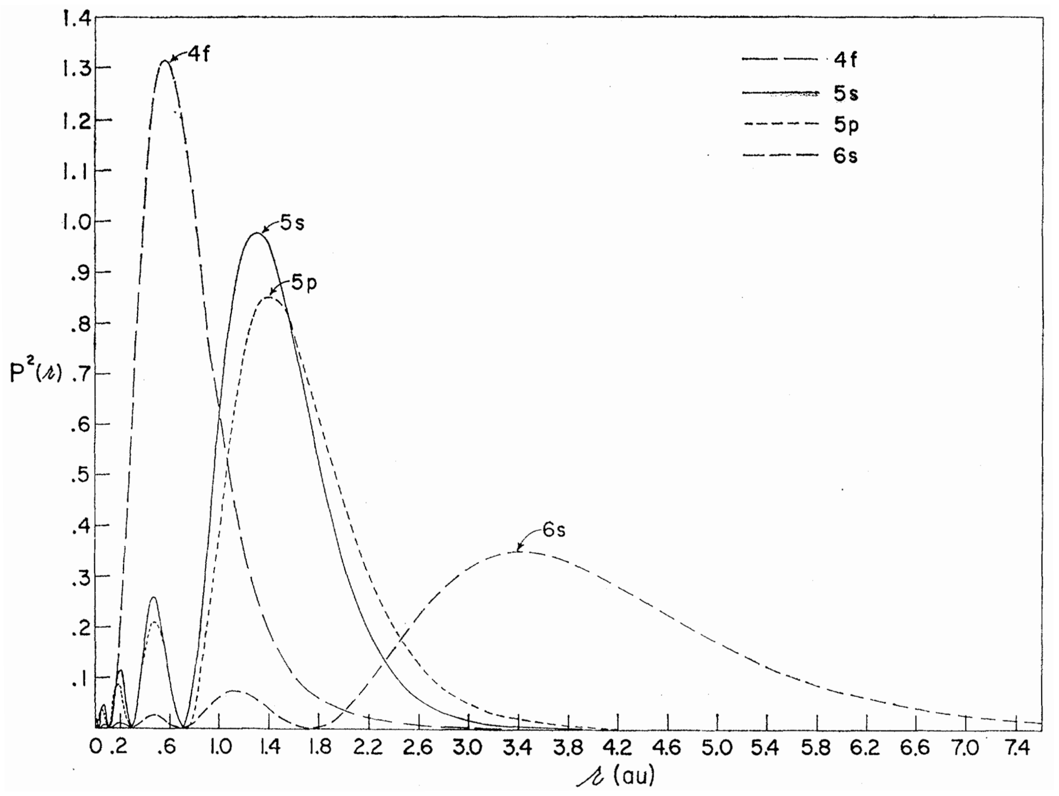


Figure 1.2: Radial probability density distributions for electrons in various orbitals of free Gd$^+$ ions, from Hartree-Fock approximation. The distributions illustrate that electrons in the partially filled 4f orbitals exist largely within the bound of electrons in the filled 5s and 5p orbitals. Reprinted figure with permission from [17], Copyright 1962 by the American Physical Society.

Though the $4f$ orbitals of the hydrogen atom are energy-degenerate eigenstates that are distinguishable with angular momentum quantum numbers, the act of including other electrons lifts this energy degeneracy due to the Coulombic interaction between the various electrons. Taking spin-orbit coupling into account as well leads to a separation of various states for which total orbital angular momentum is a good quantum number. It's important to note for future reference that the excitation of an electron should be considered a transition from one many-body quantum state to another many-body quantum state. As a result, I endeavor to refer to the state of the ion, rather than the state of the electron.

The free-ion spectra of rare earth ions and actinides are the most complex atomic emission spectra, and were accordingly the last spectra to be rigorously investigated experimentally [18] and modeled theoretically. In the early 1960s, immediately following these measurements, theoretical descriptions achieved success in both fitting parameters in Hartree-Fock approximations to measured transition energies [19, 20] as well as the independent development by Judd [21] and Ofelt [22] of an accurate description of the weak dipole transitions between what might naïvely be thought to be parity forbidden $4f$ intra-orbital transitions. These small dipole moments result in long optical lifetimes for the associated transitions and weak interaction with external radiation.

In rare earth ions, transitions between the lowest states of each of the total angular momentum manifolds have useful lifetime and coherence properties, since decay rates within each manifold can be faster. The transitions most often used in quantum optical experiments are the $^3\mathrm{H}_4$ to $^1\mathrm{D}_2$ transition in praseodymium, the $^4\mathrm{I}_{9/2}$ to $^4\mathrm{F}_{3/2}$ transition in neodymium, the $^3\mathrm{H}_6$ to $^3\mathrm{H}_4$ transition in thulium, the $^7\mathrm{F}_0$ to $^5\mathrm{D}_0$ transition in europium, and the $^4\mathrm{I}_{15/2}$ to $^4\mathrm{I}_{13/2}$ transition in erbium, where we have used Russell-Saunders notation $^{2S+1}\mathrm{L}_J$ to denote the total angular momentum [23]. Many rare earth ion phenomena and protocols were pioneered in praseodymium or neodymium, partly because of their easily accessible transitions corresponding to visible (~600 nm) or near infrared (~880 nm) wavelengths, respectively. Europium ions distinguish themselves with the longest recorded coherence time in solid state, with a decay time of six hours at 2 K for the hyperfine structure of the ground state [24]. The rare earth ion erbium, ($_{68}\mathrm{Er}$), possesses an optical transition with a long excited state lifetime (11 ms) in the wavelength range of the transparency window of glass optical fiber. Erbium-doped fiber amplifiers (EDFAs), which use erbium dopant ions as the active material for amplifying a fiber's optical mode, enable long-distance communication across all modern fiber networks [25]. If one could develop an efficient optical qubit interface within the optical telecommunications conventional band, or telecom C band, the devices could leverage the existing global optical fiber network to build a quantum network.

Though the $5s$ and $5p$ orbitals strongly shield the $4f$ transitions from external electric fields, adding the ions as dopants to a host material leads to a small variation in energy separation for ions in different host materials. This is caused by the large DC electric field resulting from the surrounding environment of ions, referred to as the crystal field. For a low-symmetry site, this can be expected to lift all degeneracies save one described by Kramer's theorem [26]. This theorem, which can be proven quite elegantly [27], states that systems with time-reversal symmetry and half-integer total spin will have at least double degeneracy of energy eigenstates.[1] In Er^{3+}, this manifests as a degeneracy between electron spin states for isotopes with no nuclear spin. An externally applied magnetic field will break time reversal symmetry and lift this last degeneracy of Kramers doublets. This sequential lifting of degeneracies via the inclusion of additional terms in the Hamiltonian is illustrated for erbium in Figure 1.3.

Rare Earth Hosts

Upon surrounding a rare earth ion with a substrate, variations in the local environment of individual ions will result in slightly different energy spacings between various electron energy levels. The envelope of the distribution of ion transition energies across an ensemble can be influenced by factors which are Lorentzian as well as Gaussian distributed, and thus can be generally described by a Voigt profile [31], which is the convolution of the two, namely

$$V(x;\sigma,\gamma) = \int_{-\infty}^{+\infty} \left[\frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}\right] \left[\frac{\gamma}{\pi\left((x-x')^2-\gamma^2\right)}\right] dx. \tag{1.2}$$

As such, it can vary from completely Gaussian ($\gamma = 0$) to completely Lorentzian ($\sigma = 0$). Furthermore, each individual emitter will have a characteristic linewidth, which we call the homogeneous linewidth. For an ensemble, the absorption profile consists of the sum of the homogeneous linewidth of each ion placed at that ion's detuning. The combination of the Voigt distribution of detunings with the individual homogeneous linewidth of ions is illustrated in Figure 1.4, and has a linewidth which we call the inhomogeneous linewidth.

Since broadening of the inhomogeneous linewidth is caused by static variations between the environment of different ions, it typically arises from variations in the crystal field due to lattice strain or defects. These causes can range in dimensionality from 2D grain boundaries and interfaces to 1D dislocations to 0D point defects. As the rare earth ion dopants themselves are point defects, high dopant concentrations also lead to an increase in the inhomogeneous linewidth.

[1]Though it's too great a digression to include in the main text, the outline of the proof involves showing that $T = i\sigma_y C$ acts as a time inversion operator, where C gives the complex conjugate. Then T commutes with a time invariant Hamiltonian, and so if ψ is an energy eigenstate, then $K\psi$ is as well, with the same eigenvalue. It is then straightforward to show that they're either the same eigenstate, or that they may differ by a sign if the number of fermions is odd (as you obtain a factor of i every time you apply T). This proof was shown first by Wigner [28], and it was recently shown eigenvectors need not exist for the theorem to hold [29].

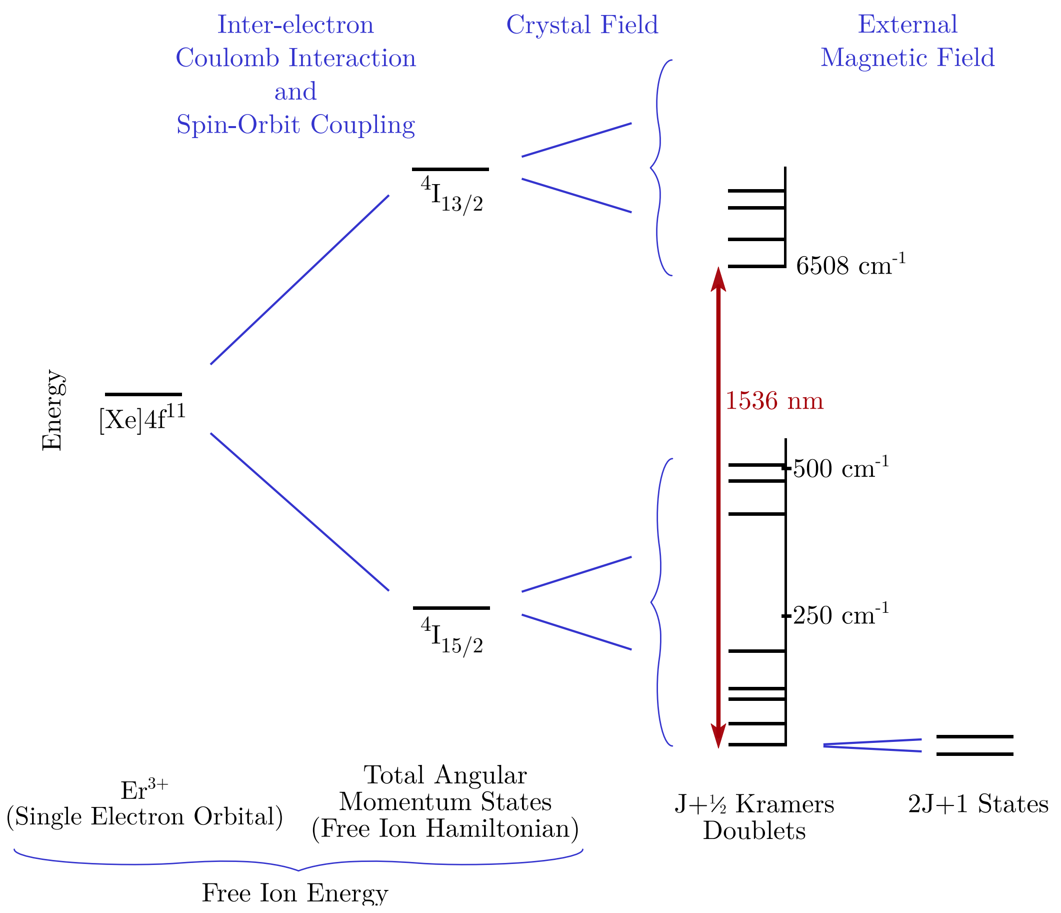


Figure 1.3: Energy levels of an erbium ion as degeneracies are lifted. Including electron-electron interaction and spin-orbit coupling leads to a partial lifting of the degeneracy of the hydrogen-like 4f orbital, leaving total angular momentum states as acceptable quantum numbers. Addition of the crystal field for a low-symmetry crystal site, like those in yttrium orthosilicate (YSO), results in the lifting of all but the degeneracy guaranteed by Kramer's theorem, which can be lifted by an applied magnetic field. For erbium ions replacing yttrium ions in site 1 of YSO, the wavelength of the transition between the lowest crystal field levels of the $^4I_{15/2}$ and $^4I_{13/2}$ state corresponds to 1536 nm light. The splitting due to the magnetic field is typically on the order of 10 GHz/T [30], depending on field orientation. In ^{167}Er, the nonzero nuclear spin results in 8 possible electron spin states which couple to the unpaired electron spin in both of the would-be Kramers doublet, lifting the degeneracy as 16 hyperfine states where even isotopes have two.

Generally, it is useful to have a low inhomogeneous linewidth, as this provides higher optical depth for a given bandwidth, and allows the spectral resolution of various transitions and increases optical depth for a given concentration. As a result, erbium and other rare earth ions are usually placed as low-concentration dopants in a host crystal containing yttrium, which has similar valance electron

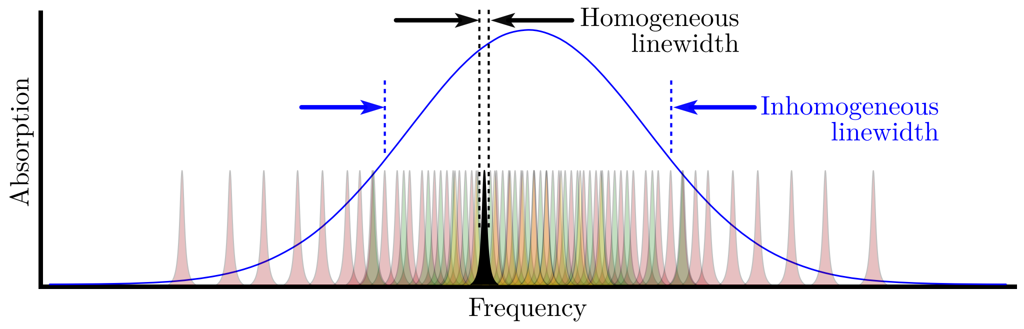


Figure 1.4: Absorption as a function of frequency illustrating the inhomogeneous linewidth in blue, which is the sum of all absorption lineshapes for individual ions (red, green, and yellow). Though both are dependent on concentration and temperature, for 50 ppm Er:YSO around 4 K, the inhomogeneous linewidth can be as low as 500 MHz [32], while the homogeneous linewidth is on the order of 10 kHz to 10 MHz, depending on magnetic field [33].

properties and a similar ionic radius as erbium. Sometimes a larger inhomogeneous linewidth can be helpful, for example when a large-bandwidth spectral feature must be tailored in the absorption profile, as in broadband AFC memories [34]. Additionally, increasing the inhomogeneous linewidth can also be used to verify that spin-spin interactions dominate decoherence [35]. However, from an engineering perspective it is generally far easier to increase a small inhomogeneous linewidth than it is to decrease a large inhomogeneous linewidth.

Yttrium Orthosilicate

Long coherence times also require reducing the sensitivity to both magnetic fluctuations and thermal vibrations (phonons) and coupling to other active defects. This reinforces the need for a low concentration of rare earth dopants, as coupling between two dopants is a source of decoherence. Additionally, the use of a host crystal with low magnetic spin fluctuations becomes important. For this reason, yttrium orthosilicate (Y_2SiO_5), abbreviated YSO, is often chosen as a host, because all three of its constituent elements are magnetically quiet relative to other hosts. The only stable isotope of yttrium, $_{89}Y$ has a nuclear spin of one half with a very small magnetic moment of -0.137 μN [36]. Of the three stable isotopes of silicon, ^{29}Si is the only one with a net nuclear spin, and it has an abundance of only 4.7%. Lastly, the only isotope of oxygen with a net nuclear magnetic moment has a natural abundance of about 0.04%.

In this host crystal, hyperfine lifetimes for praseodymium and europium dopants have been shown to be 5 minutes and 23 days, respectively [37, 38]. Furthermore, the longest coherence time to date in solid state was demonstrated using the hyperfine levels of Eu:YSO [24]. Non-isotopically pure erbium dopants in YSO have been shown to have optical coherence times longer than 4 ms [23] and inhomogeneous linewidths on the order of 500 MHz [32]. Erbium is commonly used in silica, where the optical properties are far inferior to those in rare earth host crystals; the coherence time at 150 mK was measured to be 3.8 μs with an inhomogeneous linewidth of 1.3 THz [39, 40]. In addition, recent work in ^{167}Er:YSO at high magnetic fields demonstrated a hyperfine lifetime over 10 minutes and a coherence time over a second [41].

YSO is a biaxial monoclinic crystal, characterized first by Harris and Finch [42] and later more precisely by Maksimov et al. [43]. It belongs to the C_{2h}^{6} space group and thus has fairly low-symmetry. The dimensions of the unit cell, as determined by x-ray defractometery are $\mathbf{a} = 1.04$ nm, $\mathbf{b} = 0.672$ nm, and $\mathbf{c} = 1.24$ nm, and the angle between **a** and **c** is $\beta = 102.65°$. Four unit cells are shown in Figure 1.5 (a). Yttrium ions occupy two unique sites within the crystal, both with C_1 symmetry, with one surrounded by six oxygen atoms, and the other surrounded by seven. Accordingly, dopants in site 1 and site 2 experience differing crystal fields, and thus exhibit slightly different energy level spectra, though they can be easily distinguished spectroscopically. Despite this, the transition from the lowest energy state of the $^4I_{15/2}$ total angular momentum manifold to the lowest energy state of the $^4I_{13/2}$ manifold has a wavelength near the center of the telecom C band [32] for erbium dopants occupying either site for YSO. Due to the usefulness of and focus on this transition, I refer to these two levels as the ground and excited states of the optical transition, respectively, throughout this thesis. Additionally, it is shown in Figure 1.5 that yttrium ions can fall into one of two sites linked to each other via inversion through the center of the unit cell; this results in opposing responses to an applied magnetic field unless the applied field is applied along specific high-symmetry directions.

The low crystal symmetry also results in a directional dependence of the refractive index and dipole moment. The optical indicatrix, an ellipsoid which describes the refractive index as a function of k-vector direction, does not have principal axes perfectly aligned to the unit cell directions of **a**, **b**, and **c**. For k-vectors along the principal axes of this ellipsoid, there is no observed birefringence, and thus the ellipsoid orientation can be determined optically. The three principal axes of the

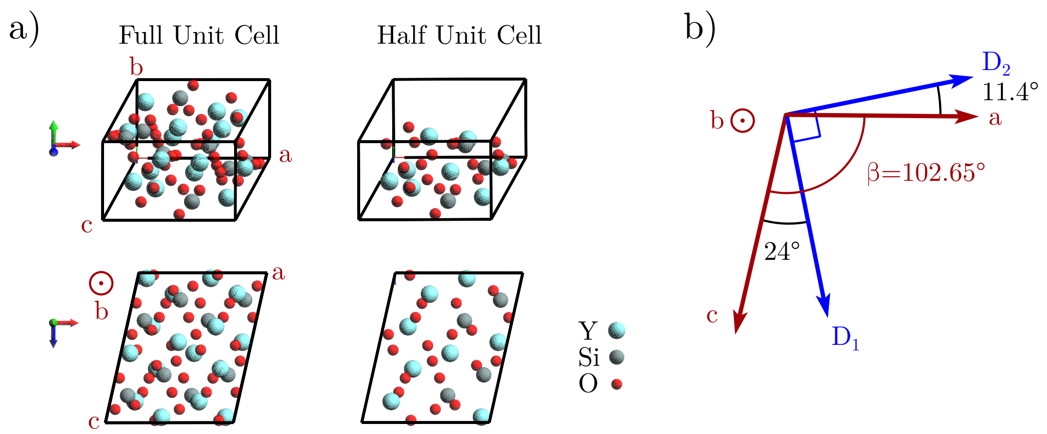


Figure 1.5: a) Illustrations of a unit cell of yttrium orthosilicate (YSO), showing two different angles for both a full and half unit cell each. Inverting a copy of the lower half of the unit cell though the center will reconstruct the full unit cell, which consists of 4(Y_2SiO_5). The low symmetry results in the lifting of most degeneracies and allowed quasi-dipole optical transitions. Image rendered using Avogadro and data from [43]. b) The orientation of the crystallographic axes **a** and **c** relative to the principal axes $\mathbf{D}_1$ and $\mathbf{D}_2$. Both sets of axes share the same **b** axis, which is oriented out of the plane here.

indicatrix have unique magnitudes, which by definition means YSO is biaxial. Due to symmetry constraints, one of the principal axes of the indicatrix must be aligned with the **b** axis, while the other two principal axes lie in the in the **a**-**c** plane [44]. Li et al. labeled these principal axes $\mathbf{D}_1$ and $\mathbf{D}_2$ [45]. Much of the literature use a right-handed variation on the coordinates set by Li et al., and have $\mathbf{D}_1$ at 24° from the **a** axis toward from the **c** axis, and thus 79° from the $\mathbf{D}_1$ axis [44]; $\mathbf{D}_1$ is in the a-c plane and perpendicular to $\mathbf{D}_1$ and **b**. These orientations are shown in Figure 1.5b. Using the measurements by Beach et al. [46] and the interpretation of the coordinates of Sun [44], we can calculate the refractive indices of YSO for a wavelength of 1536 nm to be $n_{\mathbf{b}}$=1.770, $n_{\mathbf{D_1}}$=1.769, and $n_{\mathbf{D_2}}$=1.789. The 1% difference between the largest and smallest refractive indices was taken to be negligible in most of the simulation work.

Erbium-doped Yttrium Orthosilicate

There exists a large body of work characterizing and probing the lifetime and coherence properties of erbium ions in bulk YSO, specifically measuring the aforementioned optical transition near 1540 nm. Much of this work was performed by Böttger, Sun, Cone, and Thiel. This includes precise measurement and characterization of the absorption and emission spectra and measurement of the lifetimes of the excited states for the site 1 and site 2 optical transitions [32]. The causes and rates of optical decoherence as a function of temperature and erbium concentration, as well as the applied magnetic field magnitude and direction were also explored [33, 47]. The magnetic g tensors were fit to measurements for the lowest energy states of the $^4I_{13/2}$ and $^4I_{15/2}$ manifolds for erbium ions in both site 1 and site 2 [30]. Additionally, the use of YSO co-doped with erbium and europium enabled an experiment which yielded strong evidence that spin-spin interaction dominates decoherence for low temperature and moderate magnetic fields [48], later corroborated with measurements on a scandium-erbium codoped sample [35].

Substantial progress has also been made optimizing the applied magnetic field Zeeman level lifetimes and implementing optical storage in Er:YSO by Hastings-Simon, Lauritzen, and de Riedmatten under Afzelius and Gisin, including lifetime characterization in the low magnetic field regime [49], use of auxiliary methods to improve state initialization [50], and the demonstration of optical storage at the single photon level [51]. Furthermore, evidence was presented that suggested the existence of long-lived hyperfine states for the odd isotope of erbium, ^{167}Er [49], which was later followed by identification of two lambda systems in isotopically purified ^{167}Er:YSO [52].

While there is far too much information to summarize thoroughly in a subsection of the introduction, some relevant details from the literature should be included. The experimental work in this thesis primarily uses erbium ions in site 1, which has a transition at a wavelength of 1536.45 nm and the dipole moment for this transition is largest for light polarized along the $\mathbf{D}_2$ direction [32]. The application of a magnetic field lifts the Kramers degeneracy of the lowest $^4I_{13/2}$ and $^4I_{15/2}$ states, separating a spin-up and a spin-down state for both the $^4I_{13/2}$ level and $^4I_{15/2}$. The splitting of the ground state is $g_g \mu_B B$, where g_g is the ground state magnetic coupling tensor (a rank-2 tensor), $\mu_B = \frac{e\hbar}{2m_e}$ is the Bohr magneton, and B is the applied magnetic field vector; this is illustrated on the far right of Figure 1.3. Analogously, the excited state splitting is determined using g_e, the excited state magnetic coupling tensor. The complete magnetic coupling tensors were determined completely from measurements by Sun et al. [30]. Using these, one can determine the expected splitting between both spin preserving and spin-flipping optical transitions. Furthermore, because the two g tensors do not share an eigenbasis, the relative coupling strength to the two excited states will vary as a function of magnetic field orientation for a given ground state. Similarly, decay rates from an excited state into each of the two ground states will be dependent on the magnetic field orientation. As it will be useful later, the branching ratio, β_r, is defined to be the probability of the decays from an occupied state to a specific lower energy state.

1.3 Spectral Tailoring and Echos

The measurement techniques to determine lifetime, decoherence rates, and causes can be complex and nuanced, and this description is far closer to the minimum utilitarian explanation required than a complete summary. Measurement of excited state lifetime is somewhat straightforward, while measurement of the coherence time is less intuitive and will be described in greater detail.

Spectral Hole Burning

First we begin with the measurement of excited state lifetime, which is achieved in an intuitive manner, namely setting up the system in the excited state and checking some time later to determine if it's still in that state. Rather than pump the entire ensemble, it is typical to probe the dynamics of ions using only a subset of the ions, selected using a range of transition energies smaller than the inhomogeneous line. Shining a laser on the sample and exciting this subset of ions will move the population to an excited state, from which it will decay to various lower energy states at rates described by the appropriate branching ratios and a characteristic lifetime equal to the reciprocal of the sum of the rates. For Er:YSO under an applied magnetic field, there is only one metastable ground state besides the initial state into which the excited state can decay. If the decay rate of the other Zeeman state into the initial state is sufficiently slow compared to the decay rate of the excited state, population will accumulate in the other Zeeman state. A frequency scan of the absorption will show a decrease in the absorption spectrum of the ions where the ions were resonant with the driving laser; this feature is known as a hole or spectral hole. A diagram illustrating the relevant levels, transitions, and spectral features is presented in Figure 1.6. The figure also illustrates the formation of side-holes and anti-holes, which are two modifications of the normal absorption spectrum due to the redistribution of population from the spectral hole. The frequency scan can be performed at various delays after the pump laser is turned off in order to determine rate at which the hole fills. Because the optical density is proportional to the difference in occupation between the ground and excited state, the change in occupation is proportional to the logarithm of the transmitted optical intensity. As a result, the T_1 of the upper ground level, the Zeeman lifetime, is then the $1/e$ decay time of the natural logarithm of the hole size. Symbolically, the optical density, $OD \propto N_e - N_g$ and transmission $T = \exp[-OD]$.

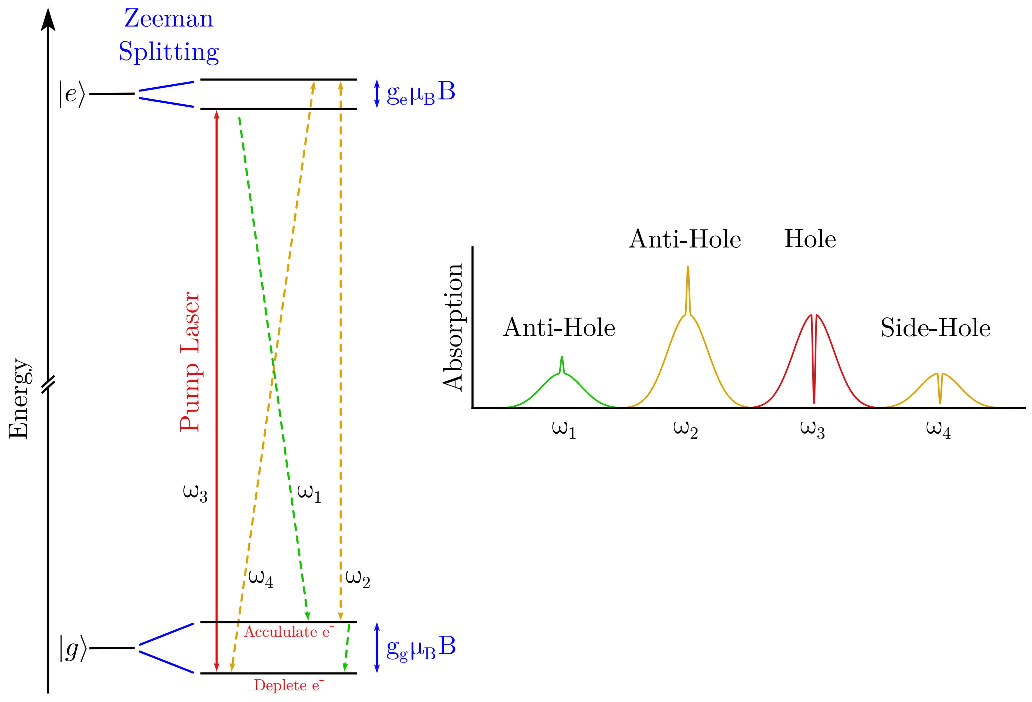


Figure 1.6: Lowest crystal field levels of the $^4I_{13/2}$ and $^4I_{15/2}$ manifolds with their respective Kramers degeneracies lifted by an applied magnetic field (left). The pump laser is tuned to frequency ω_3, exciting the transition is shown in red. Population decays along green transitions, as well as back down to the initial state. If the lifetime of the upper $|g\rangle$ state is long compared to the decay from the lower $|e\rangle$ state, population accumulates in the upper $|g\rangle$ state. Subsequent scanning of a laser will show an increase or decrease in the absorption, depending on whether there is an accumulation or depletion of population from the lower state of the transition. Transitions ω_2 and ω_3 are spin preserving, while ω_1, ω_4, and the decay between ground states involve a spin-flip and have thus been depicted by lines at an angle. A sample of this spectrum, where the pump laser is narrow compared to the inhomogeneous linewidth of the transition is shown as well (right).

The hole will fill to equilibrium as the result of a number of processes which can be classified into two categories. Namely, the hole is either filled in from the sides, or the hole is filled in by population moving from a different level. The former is known as spectral diffusion, and typically results from an interaction between spatially neighboring electron spins. The latter process is the result of decay from either the excited state or the other ground state in the Kramers doublet. It is also generally true that ions in optical excited state can and will decay to other states in the $^4I_{15/2}$ manifold, with each decay transition occurring with its own characteristic rate. Using detailed balance the individual decay processes may be dispensed with, and the multitude of individual rates can be simplified by grouping all levels other than the states of interest. This yields excited state decay rates into the ground states of interest and one rate into all other states, decay rates from the lumped other states into each of the lowest Kramers doublet, and a decay rate between the two states in the Kramers doublet.

Photon Echos

The primary tool for probing decoherence is the photon echo, which was first demonstrated in a ruby laser crystal by Abella et al. [53]. The underlying physics is analogous to the spin echo, which was first demonstrated by Hahn in 1950 [54]: both rephase decoherence in a two level system. The photon echo differs only in that energy separation between the two levels corresponds to an optical photon, rather than a microwave or radio frequency (RF) photon for spin echos.

To explain the echo itself, we return to our discussion of the Bloch sphere from Section 1.1, where the quantum mechanical state of a two-level system is represented generally by $|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle$, where $|0\rangle$ is the lower state, $|1\rangle$ is the upper state, and θ and ϕ are the angles which identify the state. As before, the latitude on the sphere, represented by θ, dictates the energy of the system, while ϕ represents the complex phase between the ground and excited states.

The time-dependent Schrödinger equation, $i\hbar\frac{\partial}{\partial t}\Psi = \hat{H}\psi$, can be easily solved for energy eigenstates, yielding $\Psi(x,t) = e^{\frac{E}{i\hbar t}}\psi(x)$, which shows that energy eigenstates accumulate a complex phase at rate proportional to their energy. As such, an excited group of ions initially on the equator of the Bloch sphere with the same phase will each accumulate phase at a variety of rates. This initialization for a photon [spin] echo is achieved by the application of an optical [microwave] driving pulse which rotates the state 90° along the **y** axis. After allowing a period of time for dephasing, denoted τ, a second application of the driving field serves to rotate the ensemble of state vectors around the **y** axis by 180°. These two pulses are called $\pi/2$ and π pulses, respectively.[2] During a second period of time of equal length, the ensemble of vectors will continue to dephase relative to each other, only now losing the relative phase difference, due to the negative sign on the already accumulated phase acquired from the rotation. After a second interval of τ, the ensemble of emitters, which

[2]Here, π and $\pi/2$ refer to the pulse area, calculated from the incident field intensity, duration, and coupling strength. For explicit computation of pulse area, as well as a wonderfully thorough discussion of two-level systems, see [55].

were previously destructively interfering, are now once more have the same phase and many will spontaneously emit. An illustration of the pulse sequence and the resulting manipulation of the state on the Bloch sphere are shown in Figure 1.7.

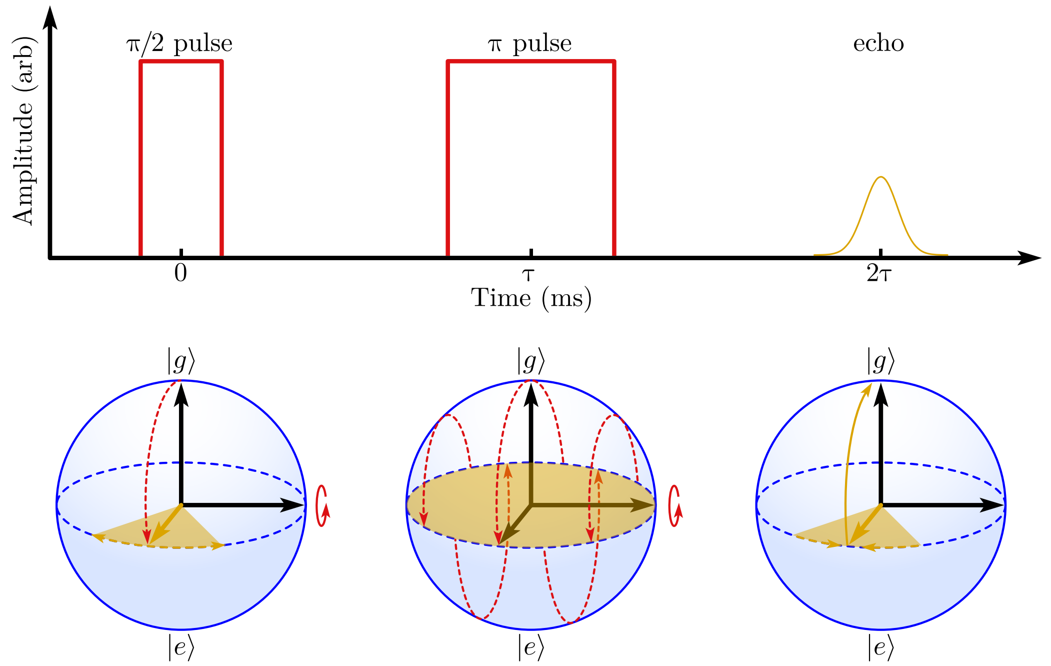


Figure 1.7: The pulse sequence of an echo and associated Bloch sphere representation of the state, where the effects of the drive laser are shown in red and spontaneous effects are in gold. First, a $\pi/2$ pulse serves to move the state vector from the ground state to the equatorial plane. The state will immediately begin to dephase. A subsequent π pulse after time τ rotates the state around the same axis, inverting the relative phase that has been accumulated since the end of the first pulse. The state continues to dephase, completing the reversal of the previous dephasing at time τ. At this point, there is no phase interference between the various components of the state, and spontaneous emission.

The canonical metaphor was presented by Hahn in Physics Today and accompanied by an image on the cover [56]. Imagine a number of runners, each with a characteristic speed, who all begin running. When signaled, at time τ after starting, all runners turn around while maintaining their characteristic speed. At time τ after the cue, all runners will reach the starting line, provided that their characteristic speed did not change. The metaphor is strained by the implication that it's possible to reverse the direction of phase accumulation. Therefore, a compromise to improve accuracy would be to imagine ants constrained to run clockwise from your perspective on a paper track; upon flipping the paper track upside down, to maintain clockwise motion the ants must reverse direction, and you would see rephasing.

If the phase is perturbed during the process, the amplitude of the resulting echo will decrease. The coherence time, T_2, is the exponential decay constant for this amplitude decay as a function of total wait time. Symbolically, the amplitude is proportional to $\exp[2\tau/T_2]$.

It's worth noting that the photon echo can be thought to be a method of storing photons, as the rephasing pulse recovers the initial pulse. However, because it relies on the coherence of the optical transition, storage time is limited compared to storage utilizing a metastable spin or hyperfine transition. Additionally, the rephasing pulse leads to inversion and spontaneous decay if the initial pulse has a low-amplitude, which precludes it as a quantum storage protocol. Lastly, to be considered an optical *memory*, the input must be able to recalled *on demand*.[3]

Many proposals and demonstrations of quantum optical memory protocols can be found, well-summarized in existing review articles [13, 57]. Of these, many are based on the dephasing and engineered rephasing of an absorbing ensemble. Below I will describe the protocol based on Atomic Frequency Combs.

Atomic Frequency Combs

Atomic frequency combs (AFCs) were shown to store light without the amplified spontaneous emission issue of photon echoes by eliminating the need for a rephasing pulse [58]. First an AFC is set up with spectral hole burning to tailor the absorption profile. Using a scanned laser frequency or pulse pairs to excite periodically spaced ions, moving population to a long-lived auxiliary state. Note that to increase optical depth (and in doing so, increase echo efficiency), population can be initialized into the ground state using the $|aux\rangle \rightarrow |e\rangle$ transition. A temporally short pulse is absorbed by the ions in the comb teeth across the broad spectral feature. Then, each of the absorbing teeth evolve with a phase. For comb teeth separated in frequency by Δ, the phases accumulated by the teeth will be $e^{E/i\hbar t}$, $e^{(E+\hbar\Delta)/i\hbar t}$, $e^{(E+2\hbar\Delta)/i\hbar t}\ldots$. Clearly, at time $t = 1/\Delta$, the phases will all match to within a factor of 2π, at which point the ensemble will re-emit the absorbed input.

Because the total storage time of the ensemble is set before it receives its input, the AFC echo described here is often regarded as a delay line. This system can be expanded to an on-demand quantum memory, however. The full proposal was first presented in [59] and demonstrated in [60]. As before, an AFC is prepared

[3]I consider this to be a somewhat arbitrary nomenclature convention, as no readout method can be instantaneous. However, there are practical, applications-based reasons why any delay between querying the memory and receiving the stored photon must be short.

and the input is sent in. Then a π-pulse is used to rotate the excitation from the upper optical state to a a long-lived spin-wave storage state. Once the pulse is to be recalled, a second π-pulse is applied, rotating the excitation back to the optical excited state, whereupon it continues its evolution for the remainder of $t = 1/\Delta$ before spontaneously emitting. Lauritzen et al. demonstrated AFC storage for 360 ns at the single photon level with an efficiency of 0.7% in bulk Er:YSO [51].

As both a long lived shelving state and auxiliary state are required in addition to the ground and excited states of the optical transition, implementing a full spin-wave memory is not possible using the erbium Kramers doublets heretofore described. It should be possible to implement a spin-wave AFC memory using the hyperfine levels of isotopically purified ^{167}Er in YSO, however, which will be addressed in Chapter 5.

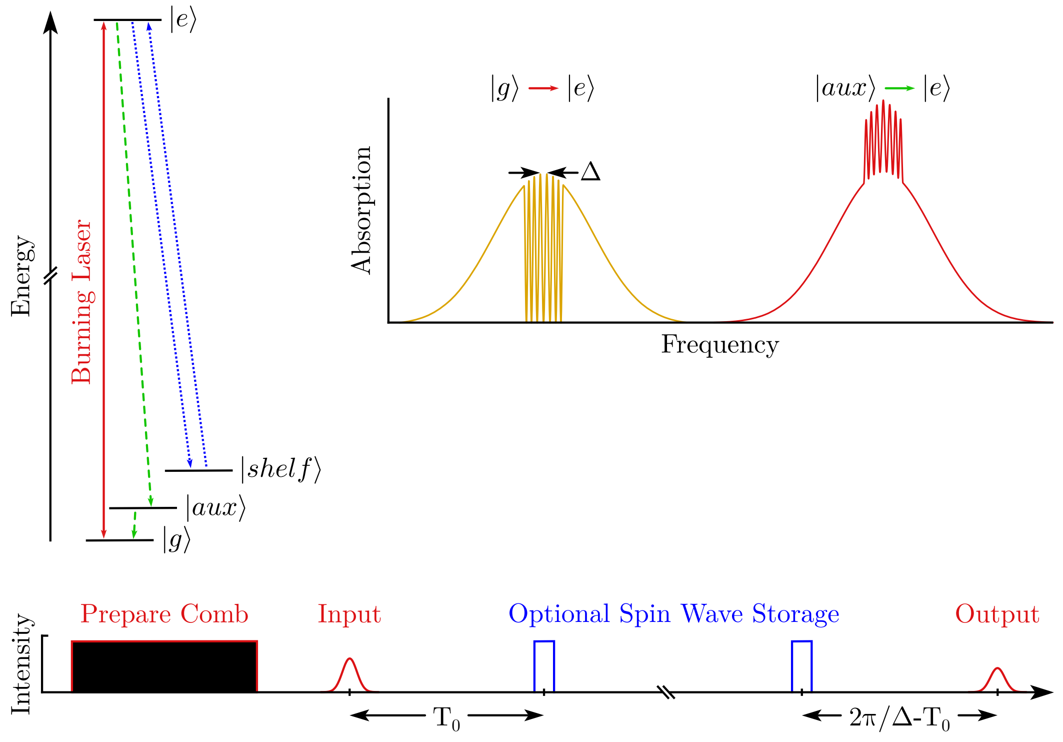


Figure 1.8: Schematic diagram of the ground, excited, and auxiliary states used in preparation of the atomic frequency comb, as well as the shelving state used for spin-wave storage. The comb on the transition between ground and excited states are burned, tailoring the inhomogeneous absorption profile. The echo is reproduced after time $2\pi/\Delta$. To implement spin-wave storage, the excitation is moved to a shelving state using π pulses (blue), which extends the storage time, limited by the coherence of the shelving state.

1.4 Rare Earth Ion Cavity Quantum Electrodynamics

At this point, outside the Faraon lab, the rare earth ion community works with ions in host crystals either in bulk, waveguides defined in various manners [61, 62, 63], or machined mesoscopic whispering-gallery mode resonators implemented a macroscopic cavity [23]. There have also been demonstrations of coupling rare earth ions in hosts to microwave cavities [64]. However, so far no optical cavities with mode volumes on the order of a few cubic wavelengths have been demonstrated elsewhere. We set out to use micro and nanoscale optical cavities to improve upon and extend this line of work, by leveraging cavity quantum electrodynamics (cQED).

In cQED, we model a system containing an optical cavity and an optical emitter, which we treat as a two-level system. The emitter's transition, with ground and excited states $|g\rangle$ and $|e\rangle$, corresponds to a frequency ω_a, which may be detuned from center of the optical mode of the cavity at frequency ω_c. We characterize the system with three rates: γ is the spontaneous emission rate of the emitter, κ is the cavity decay rate, and g is the coupling rate between the emitter and the cavity mode. The canonical cQED illustration of a Fabry-Perot cavity with the three rates is labeled in Figure 1.9.

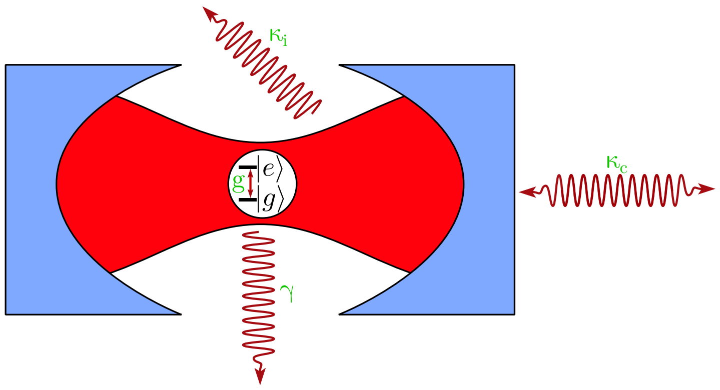


Figure 1.9: The iconic representation of cavity quantum electrodynamics, depicting a cavity (blue) with mode coupled to a two-level system (white). The system is characterized by excitation transfer rates (green), namely the coupling between the ion and the cavity, g, the spontaneous loss from the cavity, γ, and the loss from the cavity mode. Cavity loss κ is the sum of all loss rates from the cavity, which includes coupling to the input/output channel, κ_c, as well as all other loss modes, κ_i, including scattering and absorption.

Ambiguity occasionally arises in the literature surrounding whether g, γ, and κ refer to the rate of decay of the electromagnetic energy or the field; I will exclusively use the *energy* decay rates and have endeavored to adjust all equations reproduced from cited references accordingly.

The cavity is typically characterized by a quality factor and a mode volume. The quality factor is 2π times the number of oscillations sustained by the confined field before it decays to $1/e$ of its initial value, and is related to κ with $Q = \omega_c/\kappa$. The mode volume is often approximated with

$$V_{\text{mode}} = \frac{\int \epsilon |\vec{E}(\vec{r})|^2 dV}{\max\left(\epsilon(\vec{r})|\vec{E}(\vec{r})|^2\right)}, \tag{1.3}$$

where(($\vec{E}(\vec{r})$ is the electric field as a vector-valued function, max() is a function that returns the maximum value of the argument, ϵ is the permittivity, and we integrate over the position in the cavity, $\vec{r}$. It's worth noting that while this equation is exact for non-leaky microwave cavities, where the mode is confined within conducting boundaries, the equation does not truly converge for dielectric cavities [65]. However, for the low-loss cavities discussed in this thesis, Equation 1.3 is sufficiently accurate.

Increased Effective Optical Depth

A rare earth ion in bulk crystal weakly couples to its environment. While the resulting long storage time is beneficial for memories, a memory must also be able to be written to and read from efficiently, and unfortunately, this relative isolation also results in a low probability of interaction with an input photon. As a result, work performed with bulk crystals require very large ensembles of ions to attain suitable optical depth.

Intuitively, trapping the photon in the vicinity of the ion should increase the coupling, and that a smaller cavity or longer storage time in the cavity should increase the coupling. More quantitatively, if we assume the ions and cavity are coresonant at angular frequency ω, then the coupling rate for an ion at a given position $\vec{r}$ is described by

$$g_{\text{ion}}(\vec{r}) = g_0 \left| \frac{E(\vec{r})}{\max(E)} \right| \tag{1.4}$$

$$= \frac{\mu}{n} \sqrt{\frac{\omega}{2\hbar\epsilon_0 V_{\text{mode}}}} \left| \frac{E(\vec{r})}{\max(E_z)} \right|, \tag{1.5}$$

where g_0 is the coupling rate for an ion at the cavity electric field maximum, μ is the electric dipole moment of the transition, n is the refractive index at the location of the cavity maximum, $\hbar$ is the reduced Planck constant, and ϵ_0 is the vacuum permittivity. We use μ to denote the dipole moment for the given polarization of electric field. Typically the dipole moment is a vector, and the misalignment between the dipole moment and the electric field polarization is achieved by including a term $\cos(\xi) = \vec{\mu} \cdot \vec{E}/|\vec{\mu}||\vec{E}|$; however, because the $4f$ transitions in rare earth ions are not pure dipole transitions, the dipole moment cannot be expressed easily as a vector.

As multiple ions are added to the cavity, the total coupling rate, g_{tot} for the ensemble, increases with the size of the ensemble as

$$g_{\text{tot}} = g_{\text{ion}}\sqrt{N} \tag{1.6}$$

for N ions, presuming that all are coupled equally.

The cooperativity is a coupling figure of merit, and can be calculated from cQED parameters as

$$\zeta = \frac{g^2}{\kappa\gamma}, \tag{1.7}$$

where higher cooperativity corresponds to stronger coupling.[4] This can be interpreted using the definitions of the various rates such that a cooperativity of 1 implies that a photon in the cavity is equally likely to couple to the ion and back to the cavity as it is to escape to free space. Given the above expressions for g_0 and κ, we can see that the cooperativity is proportional to the ratio of quality factor to mode volume. Thus, the coupling to an ion can be increased by placing the ion in a small cavity with a high quality factor.

Increased Echo Efficiency

In the regime where the quality factor is exceedingly high, light is once again unlikely to couple to the ions, except that now the cause is the low transparency of the cavity. The cavity energy decay rate κ is the sum of the decay rates for the various channels of the cavity. It is useful to explicitly separate the coupling between the cavity and the input/output mode, κ_c, from the total decay rate for the remaining channels, κ_i. The quality factor corresponding to κ_i is known as the intrinsic quality factor, Q_i, while the Q corresponding to κ is known as the loaded quality factor, Q_l. Logically, simulations and measurements both probe the loaded Q, and cavities which are weakly coupled to the input mode will possess a loaded Q_l that nearly matches Q_i.

[4]Note that usually η is used to represent cooperativity; however, to reduce ambiguity, in this thesis η is only used for echo efficiency, described shortly.

It was shown that an optical cavity can achieve complete absorption of the input light if κ_i is negligibly small and $\kappa \approx \kappa_c$ is equal to the rate of energy absorption by the ensemble [66]. Quantitatively, the cavity-enhanced AFC echo efficiency is given by

$$\eta = \left(\frac{4\Gamma_c \kappa_c}{(\Gamma_c + \Gamma_0 + \kappa)^2} \right)^2 \exp\left[\frac{-7}{\mathcal{F}^2} \right]. \tag{1.8}$$

Here, $\mathcal{F}$ is the finesse of the AFC, defined analogously to Fabry-Pérot cavities as the ratio of FWHM to feature spacing; Γ_c and Γ_0 describe the absorption rate of light by the cavity comb and the absorbing background in addition to the comb, due to imperfect comb initialization. Perfect hole burning will reduce Γ_0 to zero, and with $\kappa_c = \kappa = \Gamma_c$, the prefactor reaches unity while the exponential term accounts for the rephasing efficiency limit due to the nonzero width of the AFC teeth. Progress toward reaching this regime, referred to as impedance matching, is discussed in Chapters 4 and 5.

Increased Hole burning Efficiency

Efficient hole burning is difficult in Er:YSO because the spin decay rate is only at most a factor of 10 longer than the excited state decay rate. As hole burning is necessary for state preparation in many optical memory protocols, creative methods have been attempted to improve the hole burning efficiency, including applying an RF field to mix the excited state population to other spin states as well as optically driving spontaneous decay to a short-lived auxiliary state higher in the $^4I_{15/2}$ manifold [50]. Coupling an optical cavity to an emitter can modify spontaneous decay rates for coupled emitters via the Purcell effect [67], allowing us to shorten the optical excited state lifetime without changing the spin state lifetime. This is demonstrated and further discussed in Chapter 4 for the purpose of improved hole burning without the application of auxiliary optical or RF fields. However, here we introduce and discuss relevant aspects of the Purcell effect.

Originally proposed for relaxation of spins in a microwave cavity, the Purcell effect describes the increase in excited state decay rate due to the interaction with a resonant cavity. The rate for any spontaneously occurring transition in quantum mechanics from initial state $|i\rangle$ to final state $|f\rangle$ is described by Fermi's golden rule and is proportional to the density of final states. A resonant cavity serves to modify the local density of final states, and we can express the ratio of the spontaneous emission rate in the cavity to the spontaneous emission rate in free space as

$$\frac{\Gamma_{\text{cav}}}{\Gamma_{\text{free}}} = \frac{3}{4\pi^2}\frac{Q}{V_{\text{mode}}}\left(\frac{\lambda}{n}\right)^3 \frac{\kappa^2}{(\omega-\omega_{\text{cav}})^2+(\kappa/2)^2}\left(\frac{\mu E(\vec{r})}{\max\left(E(\vec{r})\right)}\right)^2. \tag{1.9}$$

This ratio is known as the Purcell factor, and denoted F_{P}. The Lorentzian lineshape accounts for detuning between the emitter and the cavity, while the final term accounts for an emitter's location in the cavity mode profile. It's worth noting that when the cavity is on resonance with the emitter and spatially located at the maximum, this expression reduces to

$$F_{\text{P,max}} = \frac{3}{4\pi^2}\frac{Q}{V_{\text{mode}}}\left(\frac{\lambda}{n}\right)^3, \tag{1.10}$$

which has a clear dependence on the ratio between quality factor and mode volume. As a result, cavities with smaller mode volumes and higher quality factors demonstrate larger Purcell enhancements. If the emitter can decay via multiple channels, the branching ratio, $\beta_{\text{r}} = T_1/T_{\text{spon}}$, must be included in the above expression to account for the fact that only a fraction of the light is emitted into the cavity mode. Starting from

$$F_{\text{P}} = \frac{3}{4\pi^2}\frac{Q}{V_{\text{mode}}}\left(\frac{\lambda}{n}\right)^3 \frac{\kappa^2}{(\omega-\omega_{\text{cav}})^2+(\kappa/2)^2}\left(\frac{\mu E(\vec{r})}{\max(E(\vec{r}))}\right)^2 \beta. \tag{1.11}$$

We can substitute for g_{ion}, β, and T_{spon}[23] and thoroughly rearrange to arrive at an alternate expression,

$$F_{\text{P}} = \frac{T_1 \kappa g_{\text{ion}}^2}{(\omega-\omega_{\text{cav}})^2+(\kappa/2)^2}. \tag{1.12}$$

Thus, if g_{ion}^2 is large compared to κ/T_1, the spontaneous emission rate will be higher than that of a free ion, and the decay rate is said to be Purcell enhanced.

1.5 Outline of the Thesis

The remainder of the thesis describes progress toward implementing a scalable, erbium-based, optical quantum memories. Chapters 2 and 3 contain descriptions of the simulation, design, fabrication, and characterization of two distinct architectures of devices coupled to erbium ions in YSO. Lithography of optical cavities typically generally requires a membrane of the material to provide optical confinement, but high-quality YSO can only be grown in bulk; device architectures presented in Chapters 2 and 3 provide independent solutions to this issue. The device in Chapter 2 utilizes a short and highly serialized fabrication procedure based on focused ion beam milling of triangular nanobeam cavities. The device in Chapter 3 utilizes electron beam lithography and dry etching to achieve a scalable microcavity that can be coupled to silicon photonics. Chapter 4 discusses progress using both cavity architectures in erbium state manipulation, compares both devices, and describes progress implementing memory protocols in the triangular nanobeam cavities. Chapter 5 contains a discussion of alternative erbium systems and potential future directions.

Chapter 2

TRIANGULAR NANOBEAM DEVICES

This chapter describes the design, fabrication, and characterization of triangular nanobeam devices. While devices with this architecture have been made in the Faraon group previously for various wavelengths and using different substrates [68, 69], this chapter will focus on the YSO device which I fabricated and used to demonstrate coupling of a microscale to an erbium ion ensemble for the first time, presented in [70].

2.1 Device Simulation and Design

It has been demonstrated that light can be confined to mode volumes on the order of a cubic wavelength, using one-dimensional [71, 72] and two-dimensional [73] photonic crystal cavities. Typical high-quality-factor one-dimensional nanocavities consist of etched holes located along the center line of a nanobeam with rectangular cross-section [74]. Confinement within the beam arises from total internal reflection owing to a higher refractive index within the beam. Confinement along the beam results from a careful choice of hole area and spacing, and the periodic spacing can result in a photonic band gap wherein distributed reflection from the holes results in constructive interference among reflections. To make such a beam, the device is designed in simulations, the pattern is exposed in electron beam resist or photoresist, and the pattern defining the waveguide and holes is transferred using anisotropic etching. This process, however, requires starting with a membrane of the material to provide initial refractive index contrast with air or some substrate, and thus preventing light-leakage into the substrate.

Inspired by the work of Bayn et al. in which triangular nanobeams in diamond were milled using a focused ion beam [75], we considered removing material from two angles to create a suspended YSO nanobeam. This fabrication method was modified by Burek et al. to be scalable through the use of an angled oxygen plasma etch [76, 77]; however, this technique could not be adapted because there are no good recipes for dry etching of YSO. Thus, patterning of the YSO was to be performed using a focused ion beam.

In designing the device, we initially investigated using centered round holes, but soon found that these preliminary designs were very sensitive to perturbations in alignment of the center holes; offsetting a cylindrical hole from the center line substantially changes the volume of material removed, which changes the local effective refractive index. While alignment to an accuracy of nanometers is easily achievable in electron beam lithography, stage drift in the focused ion beam system would likely result in misalignment of holes. As a result, rectangular trenches with different widths were chosen to be milled into the top of the triangle to provide the periodic refractive index variation necessary for confinement along the length of the beam. It was predicted, and later verified, that the shape of subwavelength features would be unlikely to cause scattering from the cavity mode. The triangular nanobeam cavity's robustness against fabrication imperfections is shown in Figure 2.1. Due to difficulty calibrating the milling rate and focusing the beam, the parameters most prone to variation are the ridge depth, the top width of the nanobeam, and the ridge length. Each of these demonstrate a fairly weak influence on the cavity's quality factor and resonant wavelength.

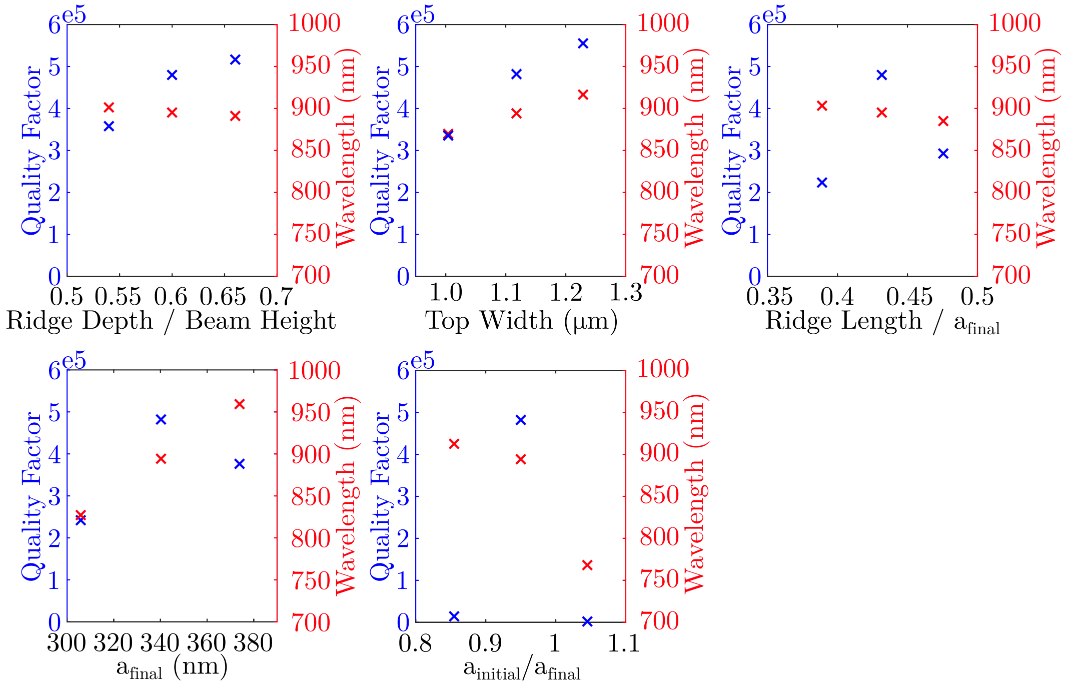


Figure 2.1: Simulation results showing quality factor and resonant frequency of the cavity mode for ± 10% errors on the five main parameters for the nanobeam resonator. Due to the fabrication method, certain parameters are more reliable to generate consistently. Parameter plots are roughly sorted from most prone to fabrication error (top row) to most consistently precise (bottom row). Simulations were run and analyzed by Alex Hartz.

Analogous to most traditional nanobeam optical cavities, we implement a constant periodicity for the “mirror” portions of the cavity and include a parabolic defect, wherein we vary the spacing between grooves to allow the cavity mode to exist in a region that adiabatically transitions to regions with photonic band gaps at the cavity frequency on either side.

A milling angle of 30 degrees was chosen to produce an equilateral triangular cross section. The cavity quality factor was optimized by searching the parameter space with finite difference time domain simulations. Simulations were performed using the freely available software package, MEEP [78]. The cavity mode is polarized in the TE direction (parallel to the top surface of the substrate and perpendicular to the beam’s length), and possesses a simulated quality factor of Q_{sim} = 70,000 and a mode volume of V_{mode} = 1.65 cubic wavelengths. The code used to simulate the beam can be found in Appendix A.2. The cavity cross-section and optical mode profiles generated by the simulation are shown in Figure 2.2.

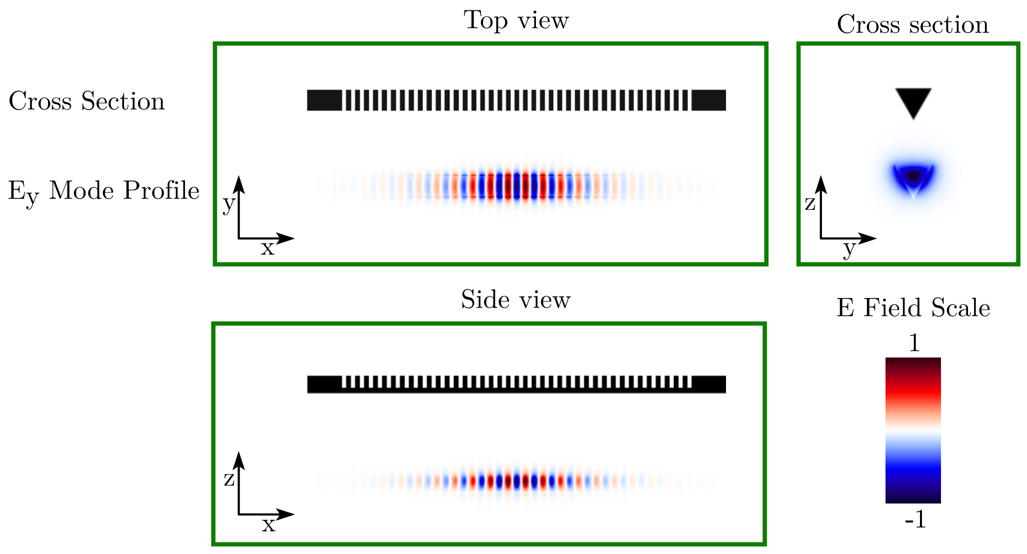


Figure 2.2: Cross sectional views of the triangular nanobeam structure and associated simulated electric field profile of the mode. The top and side show the vertical grove structure of the beam, with the smaller spacing period in the center barely visible. The transverse cross section shows the equilateral triangular shape and the TE evanescent mode on the sides of the triangle.

The triangular nanobeam optical cavities were first demonstrated with resonant frequencies at 883 nm, where they were used to demonstrate coupling to neodymium dopants in YSO [68]. To fabricate a device to couple to the transition of Er:YSO, the dimensions were scaled such that the resulting device would have a resonant wavelength of 1536 nm. For this scaled device, the top of the triangular beam was 1.38 μm wide, with 200 nm wide grooves that were 800 nm deep and spaced with a 570 nm period. The mode volume in cubic wavelengths and the simulated quality factor remain unchanged in the scaling, due to the scale invariance of Maxwell's equations in the absence of charge.

2.2 Nanobeam Fabrication

A focused ion beam (FIB) system uses magnetic lenses to focus a beam of charged particles onto a spot on a sample surface which is rastered across an area of the sample using additional steering electromagnets. This is similar to electron beam microscopy, except that in FIB systems the charged particles are gallium ions emitted from a heated and ionized source, rather than merely electrons. Our nanobeam devices were milled on an FEI Nova 600 dual beam, which images with an electron beam and mills with an ion beam that is tilted at a 52° angle relative to the electron beam.

The YSO crystal used was grown by Scientific Materials Inc. with 0.02% concentration of Er dopants, and cut such that the **b** axis was normal to the polished top surface. It was intended that the nanobeam would be oriented such that the TE mode of the resonator would be aligned to the $\mathbf{D}_2$ optical axis of the YSO crystal, because the site 1 dipole moment for the transition of interest is approximately a factor of 2 larger than in the $\mathbf{D}_1$ direction [32]. However, both of the best devices fabricated in Er:YSO to date are oriented such that the electric field is parallel to the $\mathbf{D}_1$ direction of the crystal, due to a combination of poor labeling and bad luck.

Before milling, a 50 nm layer of chromium was deposited using an electron beam evaporation tool. This layer prevents charging from either the electron beam used to image the sample, or the ion beam used for milling. Later devices use charge compensation where the electron beam simultaneously exposes the sample to counterbalance the positive charge resulting from the gallium ion beam. Chromium was chosen because chrome etchant CR-7S minimally etches YSO, and with no noticeable anisotropy on short timescales.

For all milling, the accelerating voltage on the ions was 20 keV. Before defining the beam, circular alignment marks were added on either side of the beam to mark the center of the beam along its length. The beam was then defined using two angled trenches, milled with a 2.1 nA beam. These milling steps must be performed in series, as the sample sage must be rotated between cuts to achieve the opposing angle. After each rotation step, the rectangle used to mill the subsequent trench must be manually aligned to the beam for precision. Additionally, during each step, sputtered YSO and backscattered gallium from the milling process accumulated on the underside of the beam. This material was removed using successively lower ion beam current of 81 pA. Lower ion currents allow more precise milling at a lower rate due to a lower focal spot-size. Ion currents on this device slightly differ from those later presented in [69], which were optimized for speed and precision of a beam resonant with the 883 nm transition of Nd.

The top trenches were all milled in parallel using a 81 pA beam. To calibrate the milling rate for this step, a sacrificial waveguide was used as a depth test, in which sample trenches were cut for a range of milling times to precisely determine the time required for milling to the target depth. Trench depth was calculated geometrically using the width of the waveguide spine at the bottom of the trench. Coupling trenches were milled past the end on either side of the nanobeam using a 760 pA beam, to mill a rectangle 5 μm wide and 700 nm tall at 45° from vertical.

The sample was placed in CR-7S for 40 seconds, which also was believed to remove some of the redeposited gallium and exterior YSO whose crystal structure had been damaged by the ion beam. Afterward, the sample was rinsed with DI water and blown dry with nitrogen[1]. Scanning electron microscope (SEM) images of the completed beam are shown in Figure 2.3.

It was initially a concern that some combination of gallium ion implantation, sputtering damage to the crystal lattice, or REI proximity to the crystal boundary might exacerbate decoherence rates or increase inhomogeneous broadening of the ions, but the coherence time measured in the nanobeam-coupled ions matched that measured in bulk in [68].

2.3 Confocal Microscope Characterization

The nanobeam device was characterized using a custom-built confocal microscope to determine resonant frequency and quality factor. A schematic of the measurement apparatus is shown in Figure 2.4. The optical input could be selected from multiple sources using sliding mirrors mounted onto 19 mm dovetail optical rails, labeled as "flip mirrors" in the figure. Collimated light from the input path was sent through two alignment apertures to an non-polarizing plate beamsplitter. Light

[1]Somewhat surprisingly, critical point drying was found to be unnecessary, probably due to the width of the side trenches used to define the beam.

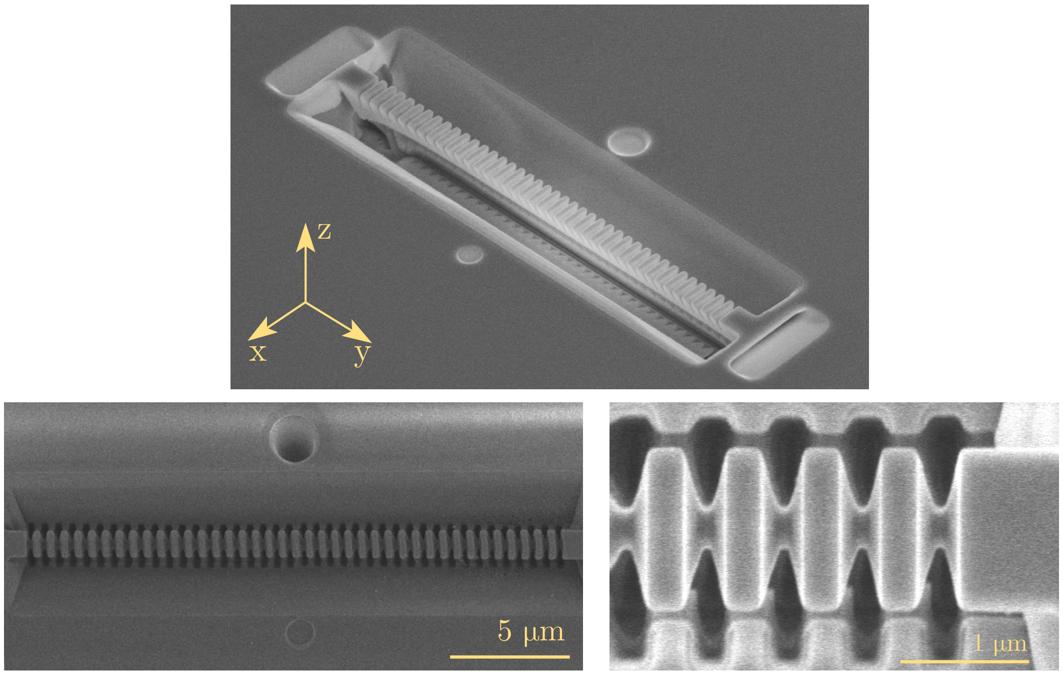


Figure 2.3: Scanning electron microscope images of the triangular nanobeam cavity in YSO. Angled trenches at the ends of the beam couple the cavity mode to free space modes used in transmission measurements. The x, y, and z axes on the image correspond to the crystal's optical axes $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{b}$, respectively. The top image was taken before removal of the chromium anti-charging layer, while the bottom images were taken after, resulting in the visible distortion (left) and brightness variations (right).

reflected by the beamsplitter was aligned to an Olympus LCPlan N 50x IR objective with a numerical aperture of 0.65 and working distance of 4.5 mm. The objective was required to focus 1536 nm light through a 1 mm of glass (cryostat window) and onto the sample. The sample containing the nanobeam cavity was indium-soldered onto the cold finger of a Janis ST-500 continuous flow liquid helium cryostat and cooled to 4.7K for the duration of the experiment. The light transmitted through the cavity was collected and collimated by the objective, and half of the light conveyed back to the beamsplitter passes to the output path. The output path was also defined by two alignment apertures, and included a pair of lenses aligned as a telescope/beam-expander. A pinhole on a flip mount was used as a spatial filter on the image plane of the telescope to selectively pass light emitted from the output port and block all other light. The collimated output of the telescope continues along the output path and was directed to various detectors using mirrors mounted onto optical rails.[2]

Input sources included a 532 nm alignment laser for easy visibility, a 1064 nm alignment laser that was visible on Si-based cameras but closer to sharing focal lengths with the 1536 light, a tunable external cavity diode laser (TUNICS PLUS SC) for narrow-linewidth scans and excitations, and a supercontinuum laser (Fianium WhiteLase Micro) for broadband transmission.

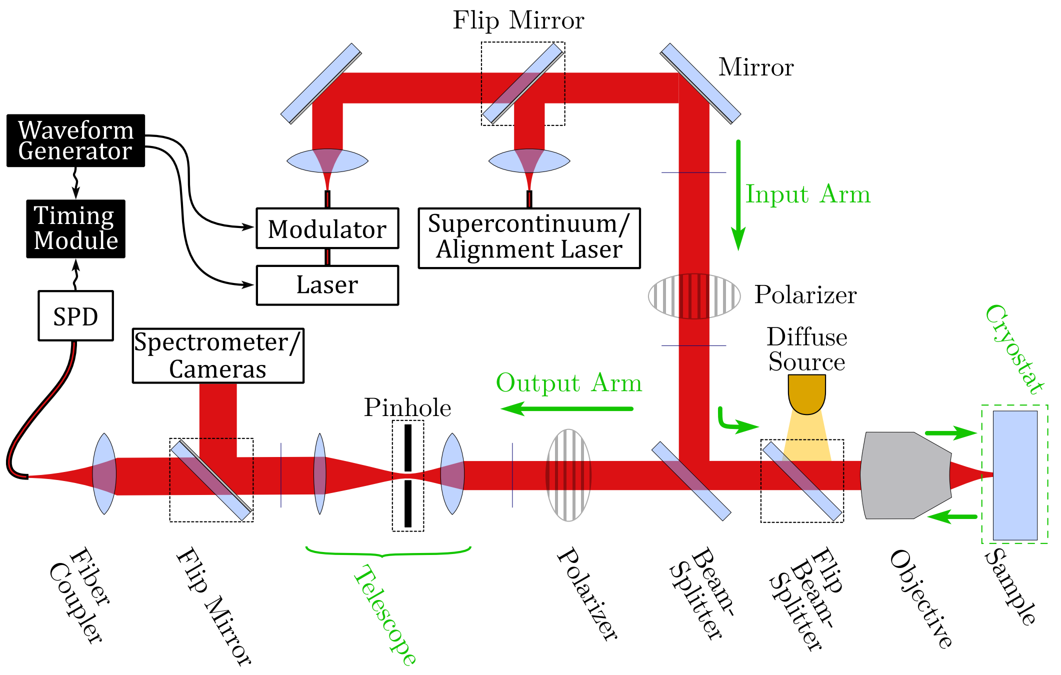
Detectors included a spectrometer (Princeton Instruments SP2750) with an InGaAs photodiode array (Princeton PyLoN IR 1024-1.7), an InGaAs/InP avalanche photodiode detector (ID Quantique ID220), or one of two cameras used for alignment. To facilitate imaging, an additional beamsplitter on a flip-mount was used to mix in diffuse light from a 940 nm (Thorlabs M940F1) LED immediately before the objective. A silicon-based CCD camera (Edmund Optics EO-0312M-GL) was used in conjunction with the 532 nm and 1064 nm alignment lasers, as well as the diffuse 940 nm light source. To facilitate improved alignment to the device, an InGaAs infrared camera (Sensors Unlimited 320HX-1.7RT) was also used to image the 1536 nm light directly.

The transmission of the nanobeam is shown in Figure 2.5. The simulated broadband transmission spectrum of the nanobeam cavity is compared to the transmission spectrum measured using the supercontinuum laser and spectrometer. A narrowband scan of the resonance is also included, achieved by frequency scanning the diode laser across the resonance and integrating the transmitted counts on the spectrometer. The quality factor of this resonance was determined to be $Q_l = 11,400$ by least-squares fitting of a Lorentzian to the transmission curve, also shown in Figure 2.5.

2.4 Cavity-Ion Coupling

Due to the small linewidth of both the cavity resonance and the ions' optical transition, fabricating a device with a cavity exactly coresonant with the optical transition upon cooling was determined to be prohibitively difficult in a reasonable amount of time. Instead, the resonance of a high quality factor device near the erbium transition was tuned to match the transition by tuning the device's resonance using the condensation of nitrogen gas [79]; the difference between the electric susceptibility of the nitrogen and that of vacuum results in a modification of the effective refractive

[2]I found optical rails had more precision and repeatability than flip mounts, which was particularly important for some components.



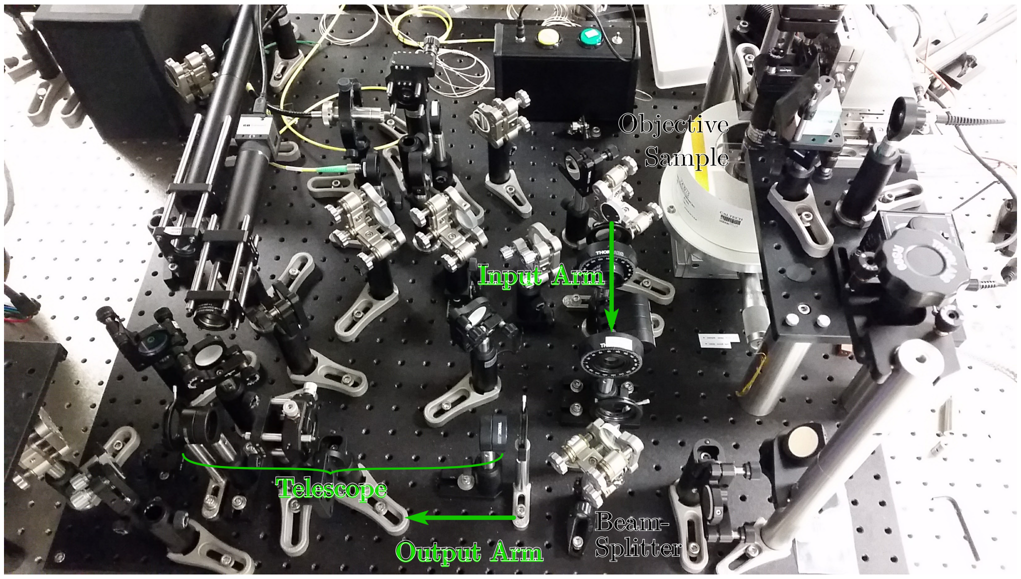


Figure 2.4: Schematic diagram and labeled image of the confocal microscope used in the characterization of the devices. Black arrows show the direction of data in cables, optical fiber is shown as a black line with a red core, and flip-mounted components are in a black dotted rectangles. Different input sources are selected using individual flip mirrors, shown at the top of the diagram. Similarly, various detectors are selected using flip mirrors on the output arm. Alignment aperture locations are depicted using unlabeled thin blue lines. The short cylindrical cryostat shown in the image is depicted in a green dotted outline.

index of the mode, which shifts the cavity mode to longer wavelengths.

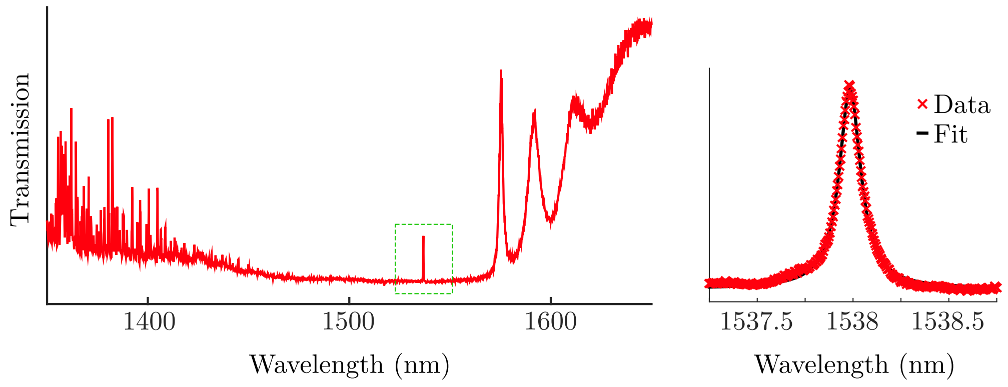


Figure 2.5: Broad and narrow transmission spectra of the detuned nanobeam at room temperature. Broad-spectrum data shows the measured transmission through the cavity using a supercontinuum laser, measured with a spectrometer, with the resonance of interest outlined in green. Inset shows a frequency scan of a narrow linewidth laser in transmission through the cavity resonance at room temperature. Fitting the transmission spectrum with a Lorentzian distribution yields a linewidth of 135 pm (17.1 GHz), corresponding to a quality factor of 11,400. A higher quality factor of $Q_l = 70,000$ was later achieved with a refined fabrication process.

The nitrogen gas was slowly bled into the chamber through a port added to the side-flange of the cryostat. On the opposite side of the flange, a 1/8” outer diameter stainless steel tube conveyed the nitrogen above the outer radiation shield and was aimed the sample. Both sides of the flange are shown in images in Figure 2.6.

A series of frequency scans of the cavity transmission are shown in Figure 2.7, illustrating the progression of tuning as the cavity resonance reaches the ion transition. Our paper presenting initial results coupling erbium ensembles to a nanobeam cavity, Reference [70], pointed out that the coresonant transmission scan at low intensity portrays a dip that is ~40% of the full transmission peak, and that using the bulk absorption coefficient for light polarized along the $\mathbf{D}_1$ direction (24.5 cm^{-1}[32]) would result in a 3.8% dip in a waveguide of the same length (26 microns). This was stated to illustrate the enhancement of coupling provided by the cavity.

It provides more insight, however, to compare the transmission of the cavity to the theoretical model of an optical cavity coupled to an inhomogeneously broadened ensemble presented by Diniz et al. [80]. Given constants a and b, and a phase accounting Fano interference between cavity transmission and a leakage mode, ϕ,

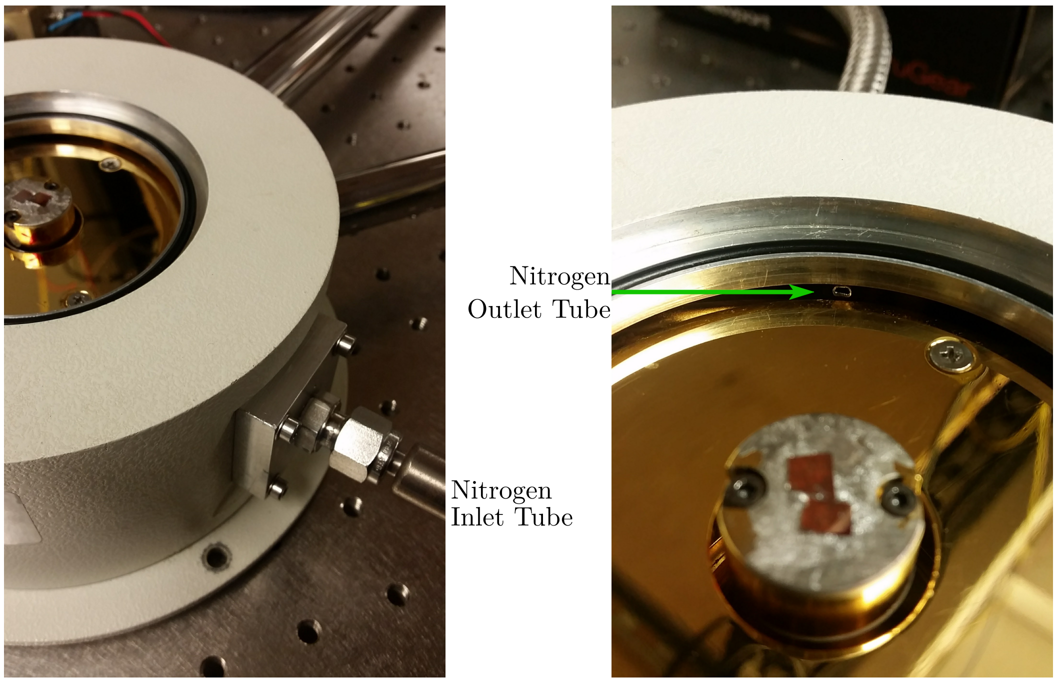


Figure 2.6: Nitrogen line into the cryostat used for condensation tuning. The tuning line meanders over the cylindrical heat shield and is slightly flattened in order to prevent thermal contact to the circular heat shield or cryostat housing. When the cryostat is under vacuum, the nitrogen travels ballistically and some scatters off of the inside of the top cover (not shown) onto the sample.

the transmission is given by

$$T = a\left| b e^{i\phi} + \frac{-\frac{\kappa}{2}}{i(\omega - \omega_{\text{cavity}}) - \frac{\kappa}{2} - iW(\omega - \omega_{\text{ions}})} \right|^2, \tag{2.1}$$

where

$$W(\omega) = -i\frac{\sqrt{\pi \ln 2}\,\Omega^2}{\Delta} \exp\left[\left(\frac{\omega + i\gamma/2}{\Delta/\sqrt{\ln 2}}\right)^2\right] \operatorname{erfc}\left(-i\frac{\omega + i\gamma/2}{\Delta/\sqrt{\ln 2}}\right) \tag{2.2}$$

accounts for the coupling to the inhomogeneously broadened ensemble, which we assume is Gaussian distributed. Here κ, γ, and ω are as previously defined in Section 1.4, Δ is the HWHM of the ensemble inhomogeneous broadening, erfc() is the complex complementary error function, and

$$\Omega = \sqrt{\int g_{\text{ion}}^2(\vec{r})\rho dV} \tag{2.3}$$

is the ensemble coupling rate between the ions and the cavity, a generalization of g_{tot} in Equation 1.6. This expression uses the ion density, $\rho = 1.87$ million ions per

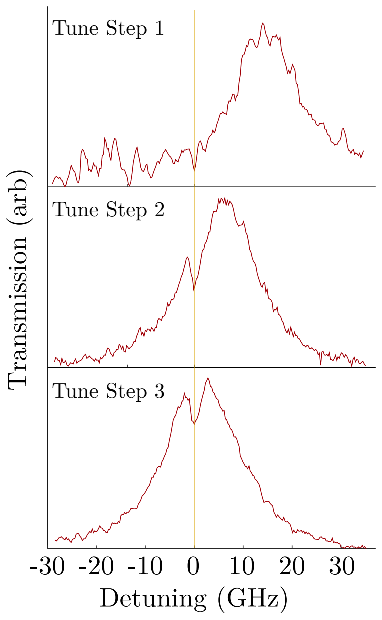


Figure 2.7: Measured transmission spectra as the nanobeam is tuned toward longer wavelengths onto resonance with the Er:YSO transition. The frequency of the transition is denoted with a thin gold line at a detuning of 0.

cubic micron,[3] and the position dependent coupling rate $g_{\text{ion}}(\vec{r})$, defined in Equation (1.4). The atomic loss rate γ is much slower than all other rates and thus neglected, and $\Delta = 0.65$ GHz is the HWHM of the ensemble inhomogeneous broadening, measured in the bulk Er:YSO.

Using Equation 2.1, we can fit the measured transmission for Ω. The result of the fit is shown in Figure 2.8, from which we can extract the value $\Omega = 1.35$ GHz. Using this, we can calculate the cooperativity, $\zeta = 0.48$, using Equation 1.7.

Once coupled to the rare-earth ions, the cavity can be used to enhance the coherent control of the ions. This will be discussed in Chapter 4, along with a detailed comparison comparing and contrasting a second device architecture presented in the following chapter.

[3]This site 1 ion density assumes 200 ppm concentration calculated using the macroscopic density of 4.44 g/cm^3 and the molar mass of 285.9 g/mol. It is also assumed that Er occupies both yttrium sites equally, due to the close match in covalent radius (Er: 189 pm, Y: 190 pm) [81].

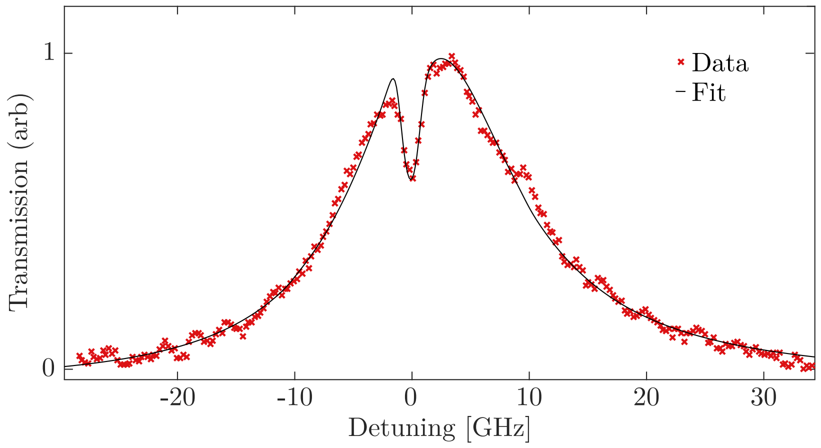


Figure 2.8: Fit of the resonant cavity transmission to the expected transmission of a cavity coupled to an inhomogeneously broadened ensemble, fitting for the detuning and ensemble coupling rate Ω.

Chapter 3

HYBRID MICRORING DEVICES

In this chapter, we describe the design, fabrication, and characterization of hybrid resonator devices where the optical mode is confined in a high index material sitting on top of YSO. The coupling to the ions occurs via the evanescent field. First I discuss some preliminary work on hybrid devices on YSO, and then I focus on the silicon photonics device which I fabricated and used to demonstrate scalable devices coupling to an erbium ion ensemble, presented in [82].

These devices were conceived to allow for scalable fabrication of multiple interconnectable, indistinguishable devices; serial fabrication of nanobeams using focused ion beam milling would be unlikely to succeed in yielding two devices which could both be tuned to the ion resonance or scaled to produce multiple qubit storage devices. Still facing the problem of requiring optical confinement within a bulk slab of material, I determined to make hybrid resonators, in which the resonators are composed of one material on an optically active substrate composed of a rare earth host crystal. These resonators would consist of a patterned layer containing most of the electromagnetic field, while the ions would couple to the evanescent tail of the mode extending into the bulk substrate. To achieve confinement and eliminate loss into the bulk, it was necessary for the material of the cavity to have a higher refractive index than the YSO substrate. As characterization equipment at 883 nm was more readily available, preliminary work focused on resonators to couple to neodymium dopants.

3.1 Initial Device Study for Neodymium Ions

Before starting fabrication of hybrid devices for coupling to Er:YSO, multiple directions were explored to develop hybrid cavities on Nd:YSO. Traditional nanobeam photonic crystal cavities, disks, and ring resonators were explored in both silicon nitride and gallium arsenide.

Silicon Nitride Devices

Both low stress silicon nitride and stoichiometric silicon nitride were found to be easily deposited on YSO as uniform films with high adhesion. However, in simulations, the refractive index of silicon nitride, $n_{Si_3N_4} \approx 2$ [83], was found to not provide enough contrast against the refractive index of YSO, $n_{\text{YSO}}{\approx}1.8$, to enable optical cavities with high quality factor and reasonably small mode volumes.

Gallium Arsenide

Unlike silicon nitride, gallium arsenide (GaAs) must be a single crystal to exhibit low optical losses, and thus cannot be deposited on an arbitrary substrate as a useful optical layer. As a result, a single crystal membrane of GaAs must be used. This is achieved by first using molecular beam epitaxy to grow a sacrificial aluminum gallium arsenide ($\text{Al}_x\text{Ga}_{1-x}\text{As}$) layer and then a GaAs membrane on top of a GaAs wafer. As the lattice constant of AlGaAs and GaAs are nearly equal [84], the added layers can be deposited in such a way that the crystal lattice is propagated perfectly during the deposition. With a room-temperature band gap corresponding to ~870 nm light [85], GaAs devices can be fabricated without substantial absorption loss for the 883 nm transition of Nd:YSO.

Since the membrane must be patterned, it is necessary to transfer the GaAs from the epitaxial wafer to the rare earth host either before or after patterning the GaAs membrane; both courses of action have their respective disadvantages. Transfer using an unpatterned membrane was attempted by selectively etching the AlGaAs in hydrofluoric acid (HF). This was found to seldom yield large planar areas, due to the fragility of the GaAs membranes. Using spin coating to apply electron beam resist resulted in visible thickness irregularities due to edge effects and wrinkles in the membrane. This, in conjunction with poor charge dissipation from the transferred GaAs membrane on insulating substrates, resulted in inconsistent exposure of the pattern. The most successful devices achieved using this method were ring resonators patterned into a GaAs membrane that had been transferred onto a glass substrate. At liquid helium temperatures, these rings demonstrated a quality factor of 24,000 near the neodymium resonance wavelength. Figure 3.1 depicts the cavity spectrum at room and liquid helium temperature, as well as the Lorentzian lineshape fitted to the cavity dip used to estimate the quailty factor. The spectra were measured by collecting the transmission through a waveguide coupled to the cavity, using a confocal microscope, analogous to the one described in Section 2.3, operating at 880 nm. A supercontinuum laser was used as the input source, and the output was sent to a spectrometer with a Pixis CCD camera.

Patterning the membrane before lift off substantially encumbers the use of waveguide coupled ring or disk resonators, since the spacing between waveguide and res-

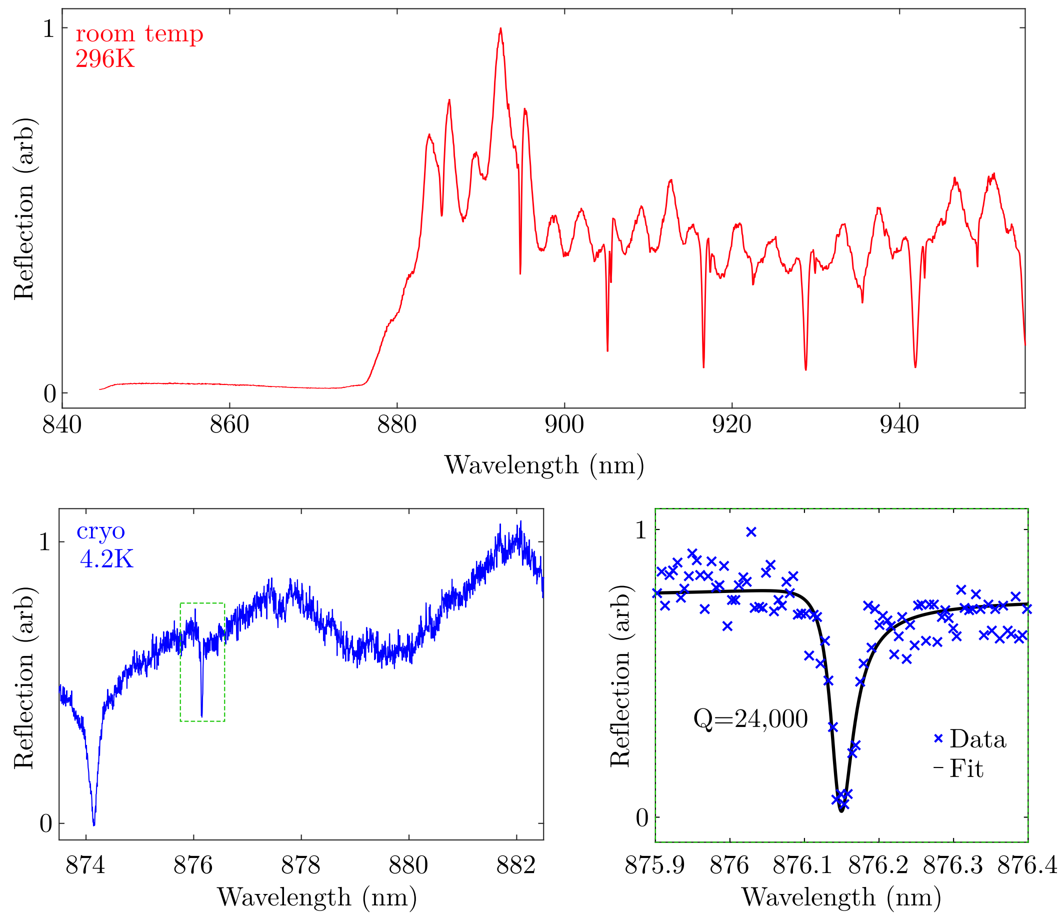


Figure 3.1: Reflection spectra of gallium arsenide rings, patterned into a membrane that had been transferred onto glass. The measured spectrum at room temperature (red) clearly illustrates the absorption onset at the band edge near 880 nm, which shifts toward shorter wavelengths at liquid helium temperatures (blue). The inset shows a fit of the lineshape on the high resolution spectrum, yielding a Q = 24,000.

onator cannot remain fixed if the membrane is fully etched. Incompletely etching the device layer, followed by transferring and then etching to release the devices was determined to be prohibitively difficult due to the low tolerance in device height and the high variability in etch rate.

Disks were fabricated on the assumption that ions coupled to the waveguide could potentially be excited, which would emit into the disk. That light might then scatter from the disk and be collected without a waveguide. Additionally, one-dimensional photonic crystals were patterned and etched, knowing that individual nanobeams could be transferred intact. To transfer nanobeams to the YSO substrate, the devices were almost entirely undercut in HF. Then, a drop of dilute HF was placed on the YSO and the GaAs device chip was placed face down on top of the YSO sample.

The sample was blown dry before the liquid completely evaporated, as YSO etches slowly in HF and evaporating HF deposits HF crystals on the substrate. The total number of devices which completed the transfer or remained on the GaAs chip were a small fraction $< 1\%$ of fabricated devices, leaving us the additional problem of improving transfer efficiency. Optical and scanning electron microscope images of gallium arsenide devices fabricated using both methods are shown in Figure 3.2.

It has since been shown that polydimethylsiloxane (PDMS) can be used to transfer mesocopic continguous sheets of patterned devices to arbitrary substrates [86]. However, this does not bypass or solve the additional issue that high quality factor GaAs devices using these two methods proved untenable even before transfer. Residual AlF_3 and $AlF(H_2O)$ [87], resulting from the HF etch of AlGaAs, found on the underside of the devices appeared to reduce the device quality factor with no readily apparent scheme to solve or circumvent this problem.

3.2 Device Simulation and Design

Amorphous silicon hydride, already being used for other devices within the Faraon group [88], was considered for use with Er:YSO as we developed optical characterization capabilities in the telecom C band. Though ellipsometry measurements found a substantial decrease in transparency for wavelengths past the visible, it was initially unclear how much absorption would be observed at 1536 nm.

As a test, mircrorings were fabricated in amorphous silicon hydride on a thermally oxidized wafer. After determining that the absorption-limited Q might be sufficiently high, microrings were designed using MEEP, optimizing the ring width and height for a first-order TM mode. Mode shape and quality factor were found to be optimized for a ring 380 nm tall and 480 nm wide. While rings with smaller radii demonstrated high losses experimentally, rings with a sufficiently large radius of 11.0 microns were found to have low bending loss in simulation and experiment

The simulated mode is depicted in cross-sections perpendicular to the vertical (z) and azimuthal (θ) directions of the ring resonator in Figure 3.3. The TM polarization was chosen to achieve an increase in the electric field at the ions. Due to the boundary condition on the continuity of ϵE for the electric field perpendicular to the surface, there is a factor of $(n_{\alpha Si}/n_{YSO})^2$ at the boundary, compared to continuity of E for the TE mode. The resulting field profile from this boundary condition is illustrated in Figure 3.3.

A high resolution simulation of the cavity mode was performed using cylindrical

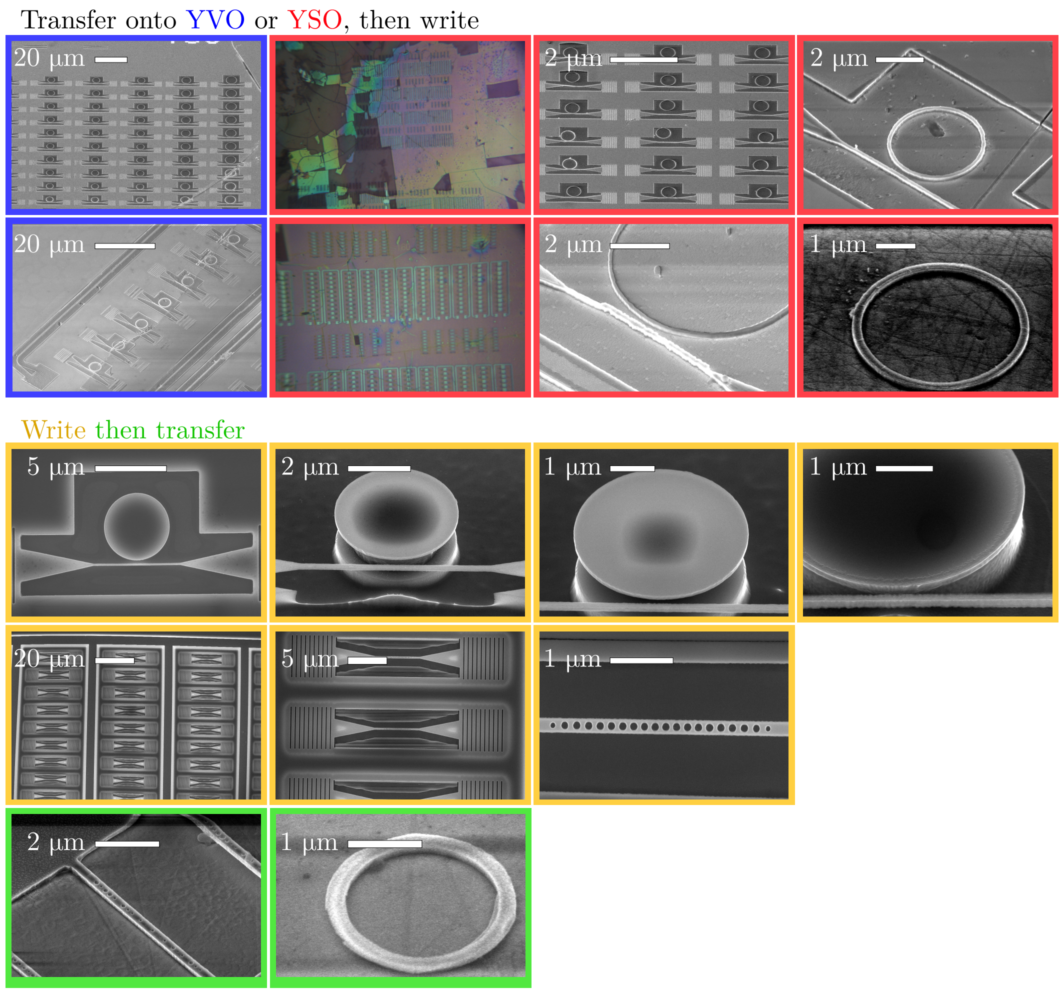
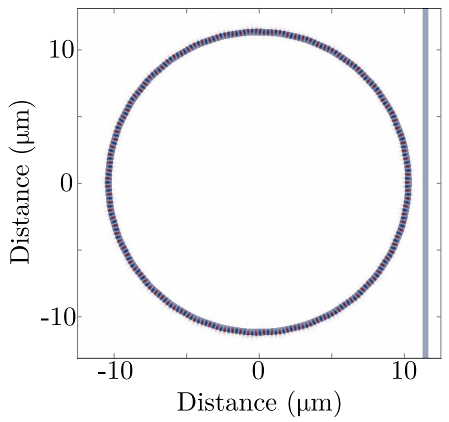


Figure 3.2: Mixed optical and scanning electron microscope images of gallium arsenide devices in various stages. The top eight images depict crystalline GaAs membranes transferred to rare earth hosts before patterning. Images with blue backgrounds are devices on YVO, in which micro-cracks in the transferred membrane are visible. On red backgrounds, the left images show patterned and developed electron beam resist before etching, viewed on an optical microscope. The rightmost four images on red backgrounds show progressive magnification of various rings on YSO, where micro-cracks, distortion, and drift in the ring location can be seen. On gold backgrounds there are waveguide-coupled disks on the top row, and photonic crystal nanobeams with parabolic defects in the second row. The images with green backgrounds show a transferred nanobeam and ring on YSO.

coordinates for a precise field distribution with low pixelation. This mode profile was used to calculate a resonator mode volume of 2.93 cubic microns. Here it is necessary to take into account the sinusoidal envelope in the azimuthal direction, which contributes a factor of 1/2. The profile was also used to calculate the coupling rates for ions, $g_{\text{ion}}(\vec{r})$, throughout the simulated volume of YSO, though in this case



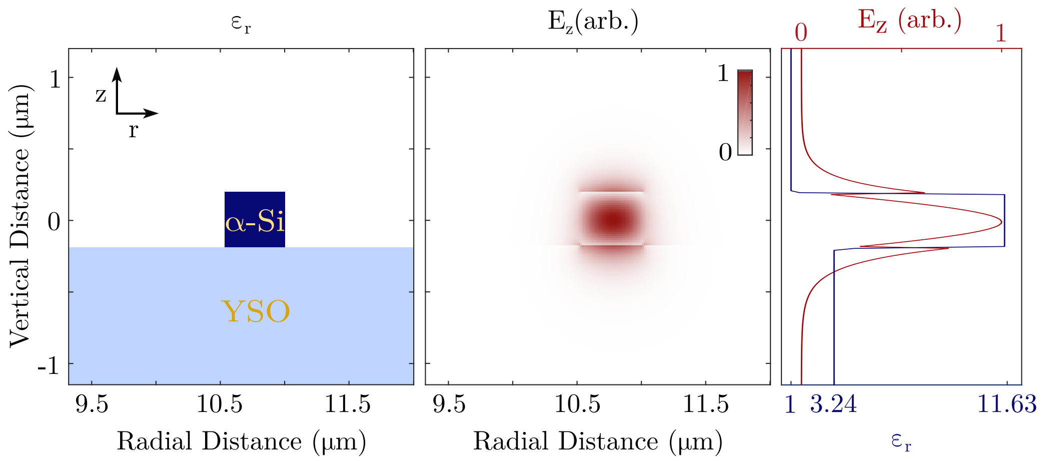


Figure 3.3: Ring resonator mode profile simulated in MEEP. The top shows a cross section of the field through the mode maximum. Below, a cross section through the ring shows the dielectric permittivity of the structure in blue (left), intensity of the electric field in red (center) and a plot through the mode maximum. The discontinuity in the field across the top and bottom of the ring results from the continuity of the perpendicular component of ϵE.

the azimuthal sinusoidal envelope is not included, as the period of the 200 THz light is negligible compared to all other timescales. As a result, on the relevant timescale of the ions, the field can be taken to be azimuthally homogeneous.

Three-dimensional simulations of the ring and a waveguide were run to determine the waveguide width and approximate the optimum spacing between the two. First, waveguide width was chosen to be slightly smaller than the ring width to better match k-vectors between the straight waveguide and curved ring. Then, simulations were run with a variety of ring-waveguide gaps; the loaded quality factors were then used to approximate the loss rate to the waveguide as a function of distance. To achieve the critical coupling described in section 1.4, the loss rate to the waveguide must match the absorption of the cavity mode by the atomic ensemble, and both must be much greater than the intrinsic loss rate of the cavity. When devices were fabricated, multiple rings were made, each with a different spacing between the ring and waveguides to allow the selection of an optimally coupled ring.

Additionally, grating couplers were simulated in two dimensions, varying width and duty cycle to maximize the amount of light scattered perpendicular to the sample. A comparison demonstrating the local maximum that was found during the optimization is shown in Figure 3.4.

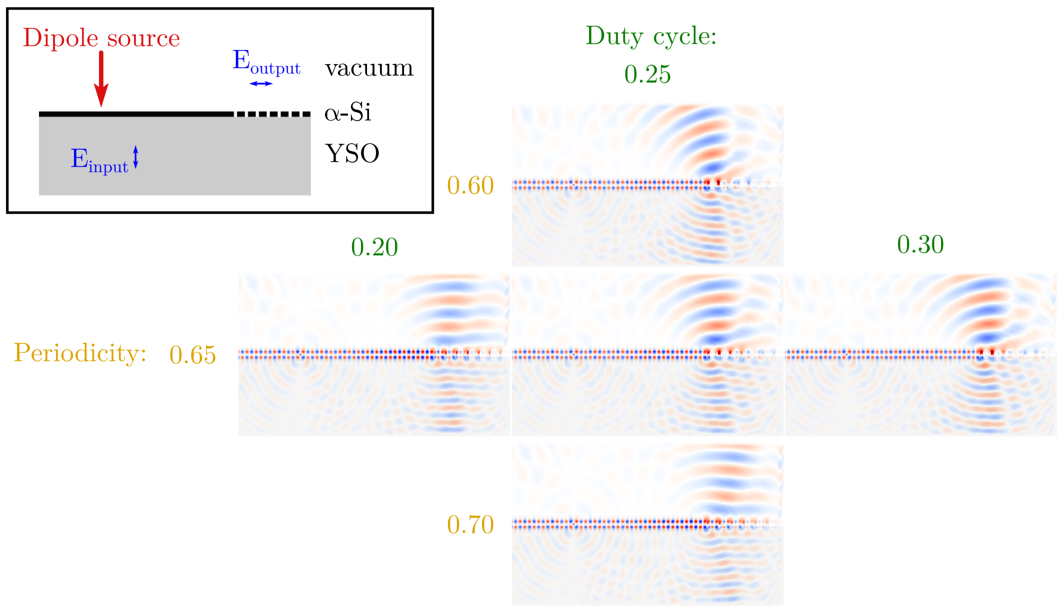


Figure 3.4: Simulations optimizing grating coupler efficiency in two dimensions. The structure is shown in greyscale in the top left corner, illustrating the location of a dipole source. To couple to a TM mode, the dipole source and depicted field profile are polarized along the directions labeled E_{input} and E_{output}, respectively. Gratings with slightly larger and smaller gratings and duty cycles are depicted on all sides, with duty cycle in green and periodicity in gold.

3.3 Hybrid Ring Fabrication

As with the nanobeam, the substrate was a Czochralski grown Y_2SiO_5 single crystal with 200 ppm erbium dopants. The fabrication is summarized in Figure 3.5 and described below.

After initially cleaning the YSO in warm Nano Remover PG (MicroChem), acetone, and isopropanol, the amorphous silicon hydride film was deposited using a plasma-enhanced chemical-vapor deposition (Oxford Instruments System 100

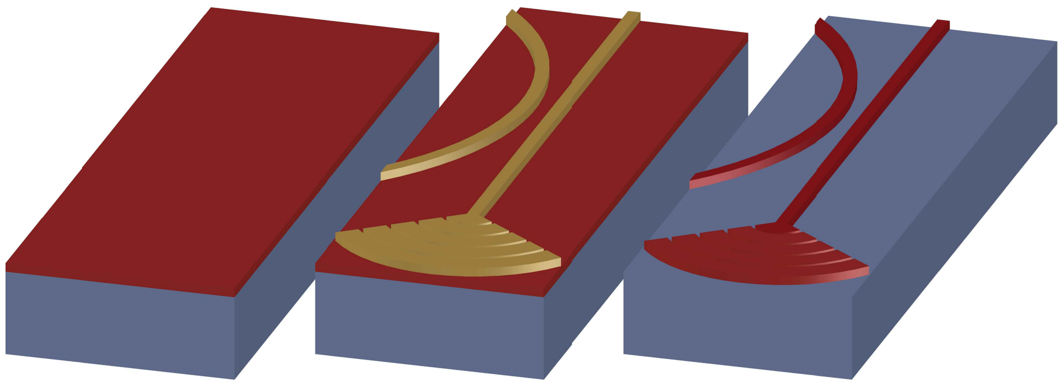

Deposit α-Si on Y_2SiO_5
Spin and pattern e-beam resist
Transfer pattern into α-Si

Figure 3.5: Overview of fabrication procedure for amorphous silicon hydride rings. YSO is shown in blue, amorphous silicon hydride in red, and electron beam resist in gold.

PECVD) using the recipe in Table 3.1. The deposition process incorporates hydrogen from the silane (SiH_4) into the silicon; as a result, the membrane is actually amorphous silicon hydride.

Etch Parameter	Value
RF Forward Power	10 W (0 W reflected)
RF Capacitors	70.3 and 20.5
5% SiH_4 in Ar Flow Rate	40.0 sccm
Chamber Pressure	801 mTorr
Temperature	200 °C
Deposition Time	21 min. 35 sec.

Table 3.1: Recipe for amorphous silicon hydride deposition on the Oxford Instruments System 100 PECVD.

Spin coating was used to apply first a 300 nm layer of electron beam resist (Zeon Chemicals ZEP520A), and then a layer of a spinnable charge-disspation layer (Mitsubishi Rayon AquaSAVE-ZX). Parameters for the spin coating steps are included in Table 3.2. As a procedural aside, it is important to let the substrate cool after baking the ZEP before applying the AquaSAVE.

The ring, waveguides, and grating couplers were then defined in the resist using a Leica Microsystems EBPG-5000+. The write itself included a bias of 50 nm and a dose of 150 μC/cm^2, with a proximity error correction for 310 nm of silicon on Y_2SiO_5. The proximity error correction was generated using the open source

material	spin speed (rpm)	ramp rate (rpm/s)	time (min.)	bake temp (°C)	bake time (min.)
ZEP520A	5000	5000	1	180	2
AquaSAVE-ZX	1500	1500	1	70	5

Table 3.2: Spin coating recipe to apply electron beam resist and charge dissipation layer to the amorphous silicon hydride layer for electron beam lithography.

Penelope, Monte Carlo simulation software [89] to generate the necessary point spread function. The resist was developed with room temperature anisole for three minutes, rinsed 30 seconds in methyl isobutyl ketone (MIBK), and blown dry with nitrogen.

The pattern was transferred from the resist into the silicon membrane using a mixed-mode etch on an inductively-coupled plasma reactive ion etch (ICP-RIE) tool (Oxford Instruments System 100 ICP 380). Prior to etching the sample, the chamber was cleaned with a high power O_2 plasma for 10 minutes, followed by a 10 minute conditioning run of the chamber. For the etch, the sample was temporarily mounted to a 150 mm wafer using Santovac 5 Diffusion Pump Oil (Kurt J. Lesker). The sample was etched according to parameters in Table 3.3. After etching, the Santovac 5 was removed using isopropanol, and the resist was removed by leaving the sample in a closed jar of Nano Remover PG (MicroChem) at 180 °C overnight.

Etch Parameter	Value
RF Forward Power	23 W (2 W reflected)
RF Capacitors	17.4 and 49.0
ICP Forward Power	1200 W (13 W reflected)
ICP Capacitors	45.5 and 58.7
DC Bias Voltage	79 V
SF_6 Flow Rate	15.0 sccm
C_4F_8 Flow Rate	40.0 sccm
Chamber Pressure	11.0 mTorr
Temperature	16 °C
Etch Time	5 min. 20 sec.
Helium Backing Pressure	4.0 Torr
Helium Backing Flow	13.4 sccm

Table 3.3: Etch recipe for amorphous silicon hydride hybrid rings on the Oxford Instruments System 100 ICP 380

The completed device is shown in scanning electron microscope images in Figure

3.6. In addition to a ring and a straight waveguide terminated in grating couplers, the device can be seen to additionally posses additional features. A second waveguide allows measurement of the cavity in transmission, collecting resonant light, which facilitates measurement. The ring can be measured as a narrow-band add or drop filter by selecting the input coupler to use in conjunction with the output coupler labeled in Figure 3.6. To increase signal-to-noise during measurement of the ring, the waveguides each have a 90° bend between the input grating and the output gratings; as a result, TM-polarized light impinging on one grating is polarized perpendicularly to TM-polarized light emitted from the output grating, and can be filtered with a polarizer. The grating opposite the bottom input grating terminates in a sharp point to scatter light and minimize the interference pattern at the opposite grating resulting from reflection. Lastly, a triangular feature between the input and transmission output grating prevents light from scattering into the slab mode directly between the input and output couplers.

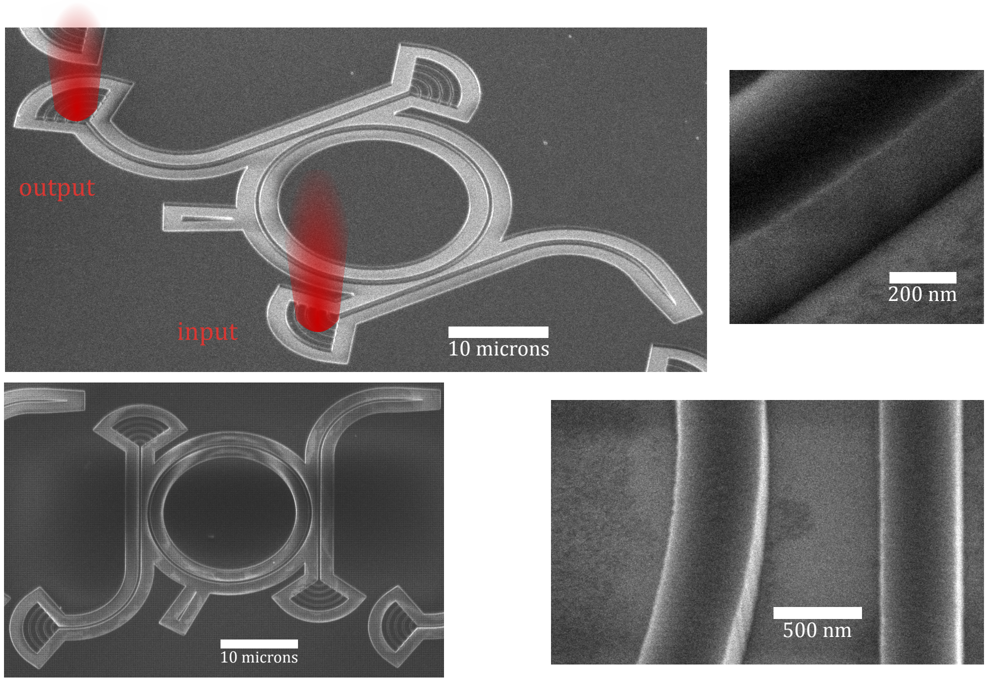


Figure 3.6: Scanning electron microscope images of the hybrid rings. The top right image includes a label of the input and output couplers for measuring transmission through the ring. Charging of the ring and waveguides results in some distortion in the bottom left image. A portion of the ring (top) and a ring-waveguide gap (bottom) are shown in the right two images. The ring can also be probed as a drop-filter by sending input through the unlabeled grating coupler using the same output grating coupler.

3.4 Device Characterization

The device was mounted and tested in the same custom-built confocal microscope used to measure the triangular nanobeams described in Section 2.3 and depicted in Figure 2.4, with minor component substitutions. Discussion of these substitutions and the results of the characterization follow.

Confocal Microscope Characterization

When characterizing these devices, a more stable diode laser (Toptica CTL and DLC Pro) was used for narrow bandwidth measurements instead of the TUNICS laser. Additionally, tungsten silicide (WSi) superconducting nanowire single photon detectors (SNSPDs) from Matt Shaw's group at JPL were used for single photon detection [90]. In addition to the pinhole used to spatially filter output light, linear polarizing filters were used on the input and output paths to allow improved extinction between the light that coupled to the ring and backscattered light.

The cavity resonances were measured in transmission using the spontaneous emission from an EDFA and measured on the spectrometer for broad-spectrum scans, and for narrow scans the diode laser was tuned across the resonance using external control of the internal piezo motor and detected on an InGaAs transimpedance amplified photodetector (Thorlabs PDA10CS). To calibrate the piezo scan, a phase electro-optic modulator (EOM) was used to add sidebands with known frequency offset which is convolved with a narrow spectral feature like the cavity resonance, which provides a conversion from scan time to optical frequency. Data from both types of scans are shown in Figure 3.7. A Lorentzian distribution was fit to the cavity linewidth with a 1.4 GHz FWHM, corresponding to a quality factor of Q_l=112,000. We believe the loss in the ring is dominated by free-carrier absorption, as the quality factor was found to increase with decreasing temperature, starting at 64,000 at room temperature.

Cavity-Ion Coupling

Tuning the cavity resonances to match the $^4I_{15/2}$ to $^4I_{13/2}$ transition of the site 1 erbium ion ensemble was performed using the same nitrogen condensation described in Section 2.4. The tuning process is illustrated by a series of cavity scans in Figure 3.8. After repeatedly tuning and detuning, by condensing nitrogen and warming to room temperature, the quality factor of the device was found to be degraded; we believe this to be the result of water adsorption to the surface because the initial quality factor was recovered by heating the device to 200 °C under vacuum.

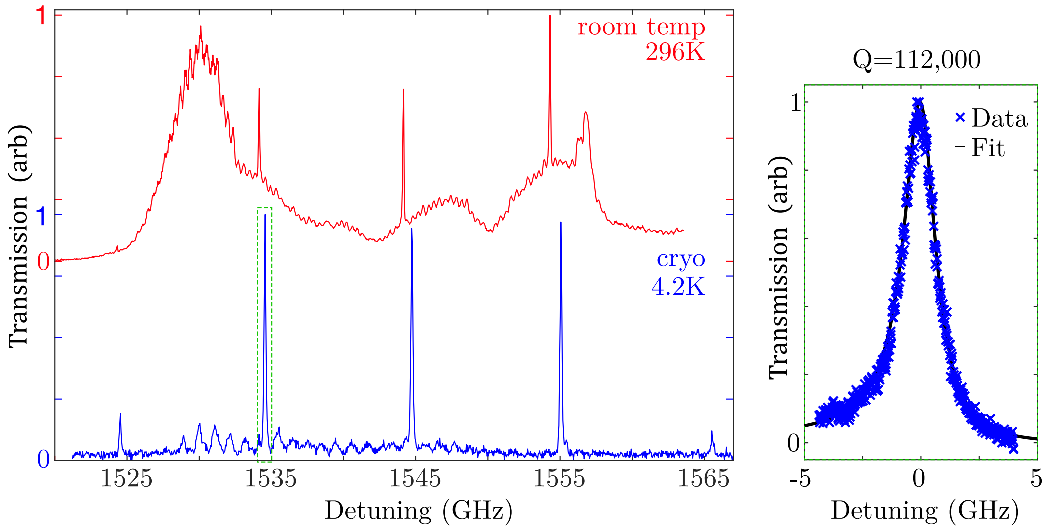


Figure 3.7: Transmission spectra of the hybrid ring at room temperature and cryogenic temperatures for a broadband scan, as well as a narrow-linewidth scan of the resonance near 1536 nm outlined in green. Quality factors and coupling strength increased as the devices were cooled, resulting in a substantial reduction in the background noise from the EDFA broadband spectrum visible in the room temperature spectrum. The resonances seem to have shifted slightly toward longer wavelengths because cooling results in a thermal contraction shifting modes slightly less than two free spectral ranges.

However, for demonstrations of cavity-ion coupling, data was taken while the Q was degraded to 80,400. This quality factor was measured and fit while the cavity was detuned from the ion ensemble. This quality factor corresponds to a cavity decay rate $\kappa/2\pi$ of 2.43 GHz.

Using the high resolution 2D simulation of the cavity described in Section 3.2, we calculate g_{ion} calculated using Equation 1.4. After discretizing the integral in Equation 2.2 and evaluating across our pixels, we find that $\Omega = 0.537$ for this cavity. We also have the HWHM of the inhomogeneous broadening, measured in bulk to be $\Delta = 0.65$GHz. Using these values, we can calculate the expected transmission using Equation 2.1, fitting only for cavity detuning from the ions; the result of this is shown in Figure 3.9. As remarked in the relevant publication [82], if the transmission data is fit using Equations 2.1 and 2.2 to determine the inhomogeneous linewidth, the resulting HWHM is $\Delta = 0.687$ GHz. The small deviation from the bulk measured value of $\Delta = 0.65$ GHz indicates that processing does not substantially alter the crystal structure of the YSO, and is unlikely to affect the optical properties.

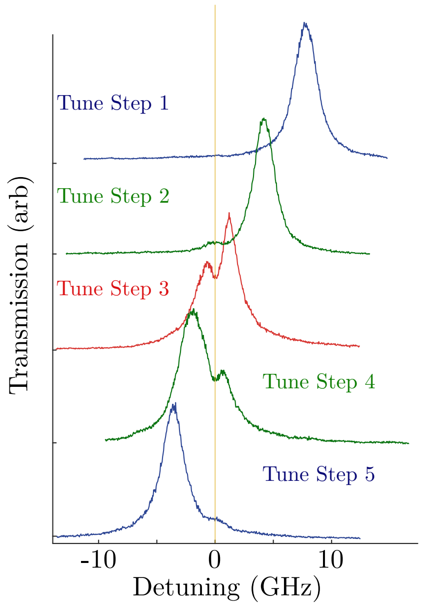


Figure 3.8: Measured transmission spectra as the microring is tuned toward longer wavelengths onto resonance with the Er:YSO transition. The frequency of the transition is denoted with a thin gold line at a detuning of 0. Due to the high quality factor, the cavity resonance appears much narrower relative to the inhomogeneous dip when compared to the nanobeam tuning in Figure 2.7

3.5 Amorphous Silicon Hydride Absorption at Shorter Wavelengths

The ease at which amorphous silicon hydride can be deposited on an arbitrary substrate makes it an appealing option for coupling to various emitters. To further the consideration of this technology for other emitters like neodymium, other rare earth ions, or color centers in silicon carbide, rings were also tested to determine compatibility at other wavelengths.

There exists a substantial body of literature describing the optical and electronic properties of silicon hydride, summarized well by Gaspari [91]. In this work, he describes that absorption onset does not begin abruptly at a cut-off frequency due to the gradual band edge of amorphous silicon hydride; rather, an effective band gap energy between around 1.7 and 1.9 eV can be extrapolated from measurements, depending on hydrogen content, though absorption is still observed at longer wavelengths due to dangling bonds. These energies correspond to a wavelength range of 650-730 nm, and thus we believe the dangling bonds to be responsible for the temperature-dependent absorption at 1536 nm.

With no adjustments to the device or confocal microscope, the ring described in Sections 3.2 to 3.4 was measured using the supercontinuum source and spectrometer

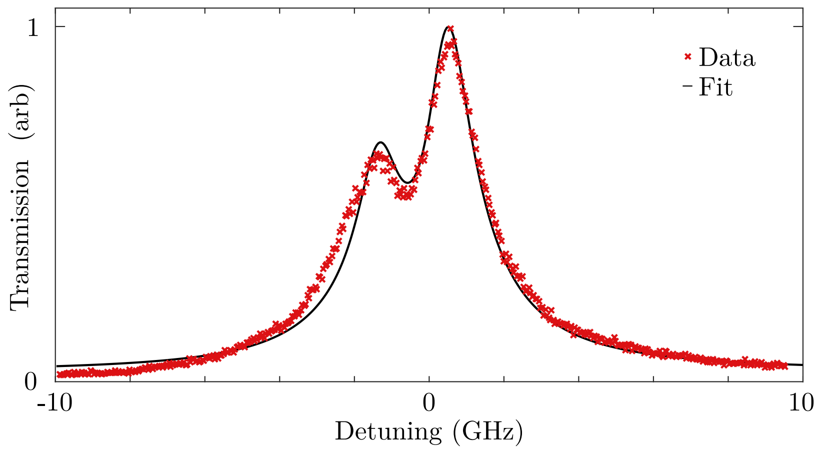


Figure 3.9: Comparison of the resonant cavity transmission to the expected transmission of a cavity coupled to an inhomogeneously broadened ensemble, fitting only for the detuning between the cavity resonance and the ion ensemble, and using measured or simulated values for coupling rates and cavity linewidth.

and demonstrated resonances from wavelengths of 1240 nm to 1620 nm, as shown in Figure 3.10. The inset of the figure shows the quality factor for the resonance at at 1270 nm is 21,000 at room temperature, which is likely limited by absorption in the amorphous silicon hydride. The resonances below 1240 are not detected due to a decrease in the grating coupler efficiency, as the ring itself should still support higher order modes at shorter wavelengths. No resonances above 1620 are detected due to the low quantum efficiency of the spectrometer camera used.

Figure 3.10 also includes transmission spectra of amorphous silicon hydride rings on Nd:YSO that have been scaled in all dimensions by a factor of (883 nm / 1536 nm) to be resonant with the $^4I_{9/2}$ to $^4F_{3/2}$ transition in neodymium. Low-resolution scans show modes of a ring spanning 40 nm. High resolution scans show multiple resonances near the transition wavelength for different sized rings. One of the resonances is shown fitted by a Lorentzian with a linewidth of 65.0 pm, which corresponds to a quality factor of 14,000. This is approximately a factor of 5 lower than the room temperature quality factor of the rings on Er:YSO, and likely results from absorption. Cooling the device to lower temperatures should mitigate this loss, as was observed in the ring on Er:YSO. However, at this point in time, silicon hydride rings did now offer sufficient advantages to switch from existing triangular nanobeams in Nd:YSO.

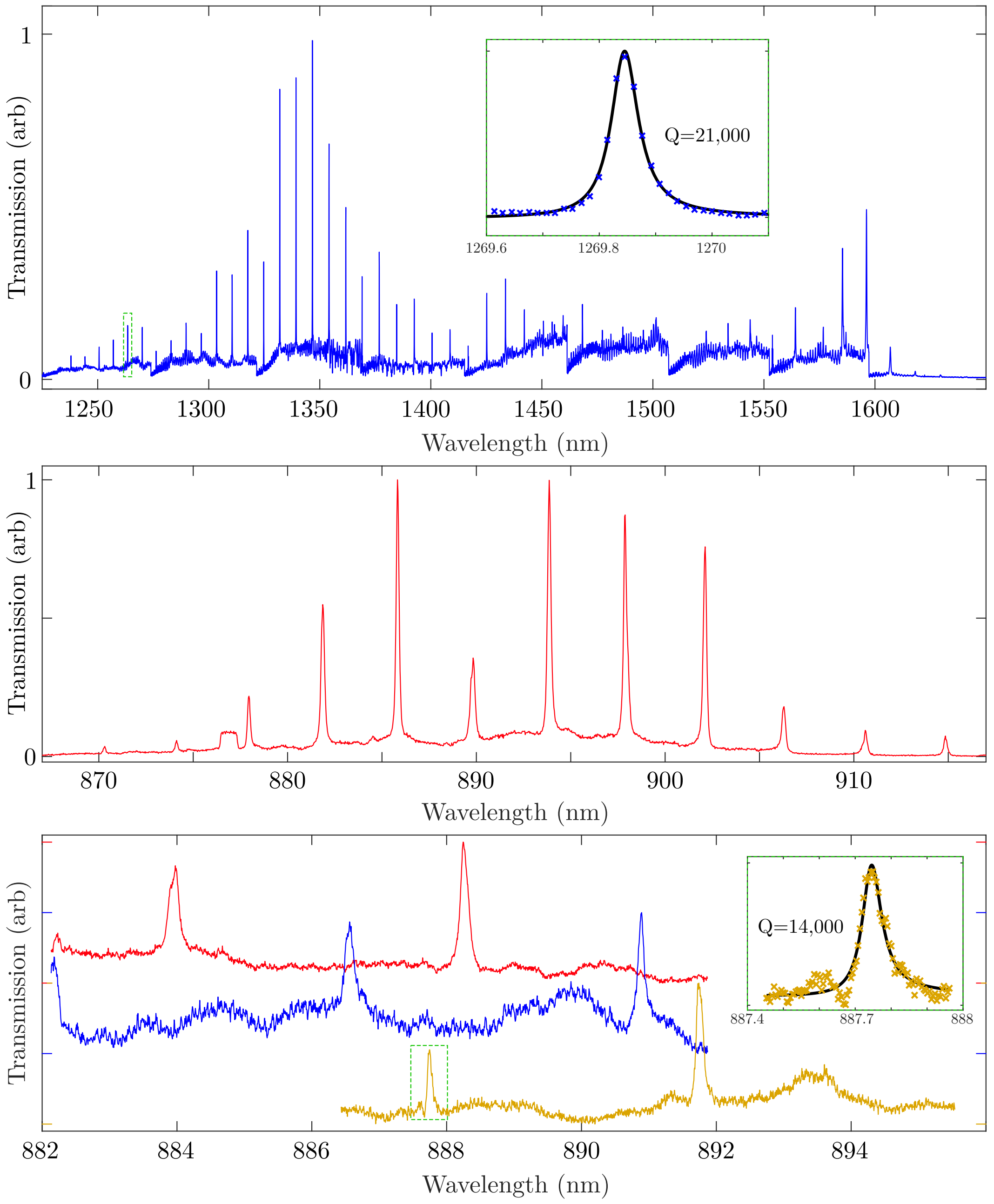


Figure 3.10: Spectra demonstrating low losses in amorphous silicon hydride across the near infrared at room temperature. The top plot shows a broad spectrum transmission of the ring described in Sections 3.2 to 3.4, demonstrating resonances from wavelengths of 1240 nm to 1620 nm. The sawtooth background noise is an artifact from the stitching together of multiple spectra. The middle plot shows a broadband spectrum of a scaled ring on Nd:YSO for coupling to the 883 nm transition in Nd shows multiple resonances, while the bottom plot shows spectra of three different rings of slightly different sizes. Resonances outlined in green were fitted with a Lorentzian to determine quality factor.

Chapter 4

OPTICAL MANIPULATION OF ERBIUM IONS

This chapter presents progress towards manipulating the state of erbium ensembles using both cavity architectures previously described. Additionally, a comparison of the two devices is provided, and the chapter concludes with a discussion of progress towards manipulating the population in the ground states of the ensemble coupled to the triangular nanobeam cavities. While the demonstration of Purcell enhanced decay rates is described in our publications on the nanobeam [70] and ring cavities [82], later material in the chapter includes unpublished work performed recently.

4.1 Purcell Enhanced Decay Measurement

For both devices, the strength of the coupling between the ensemble and the cavity was estimated using simulation and verified experimentally using the Purcell factor. While the same confocal microscope described earlier was used in characterizing both devices, input sources and output detectors were upgraded between initially measuring the nanobeam and measuring the hybrid rings.

Measuring Photoluminescence

When measuring the resonators, each cavity was tuned through resonance in steps using nitrogen condensation. At each detuning, the spectrum was measured by scanning the narrow laser across the cavity linewidth and recording the transmitted power. For the nanobeam, the output power was recorded on the spectrometer's camera as the laser stepped across the cavity linewidth,[1] When measuring the ring, the transmission of a time-resolved frequency scan of the laser was measured on the InGaAs transimpedance amplified photodetector and multiple scans were integrated on an oscilloscope. Examples of these were shown previously in Figures 2.7 and 3.8. The capability to average many fast and repeatable scans of the laser resulted in a reduction of noise for the scans of the ring resonator.

In addition to measuring the transmission spectrum at each step, the ensemble was resonantly excited with a laser pulse, and the photoluminescence (PL) was

[1]Even for the relatively low quality factor nanobeam, the cavity linewidth (FWHM = 135 pm) was too narrow for the spectrometer's resolution (46 pm/pixel) using a 600 grooves/mm and SP2750 spectrograph.

measured using single photon counters, though different amplitude modulators and detectors were used when characterizing the two devices. A fiber-coupled electro-optic modulator (EOM) was used to generate 20 ms pulses to excite the nanobeam ensemble, while 100 μs pulses were generated using a fiber-coupled acousto-optic modulator (AOM) when measuring the hybrid rings. The output of the nanobeam was coupled to a multimode fiber and measured using the InGaAs/InP avalanche photodiode detector (ID Quantique ID220) single-photon counter. The output of the hybrid rings was coupled into a single-mode fiber with a decrease in the numerical aperture from the pinhole to the fiber to reduce spurious interference patterns arising from different points on the non-negligible area of the grating coupler. This fiber conveyed the signal to a WSi SNSPD in a shielded mu-metal box on the 400 mK plate of a BlueFors helium-3 refrigerator. The optical connections and components inside the refrigerator are further discussed in Section 4.3

For the ring resonator, the frequency of the maximum of the cavity transmission was measured on a Burleigh WA-1600 wavemeter. This was used in tandem with the EOM calibration described in Section 3.4 to accurately determine the frequency axis of transmission scans. Additionally, the wavemeter was used to accurately tune the laser in order to measure PL at three frequencies, spaced by approximately a GHz in the inhomogeneous line. Detuning frequency for the nanobeam was determined using the internal calibration of the laser, which was sufficiently accurate, given the cavity's large linewidth.

Figure 4.1 shows a plot comparing bulk photoluminescence emission to emission from ions while the cavity is resonant with the ion transition frequency for the nanobeam. The figure also includes a comparison between emission from an ensemble coupled to the hybrid ring cavity and an ensemble only coupled to a waveguide. Because the local density of optical states is modified by the waveguide, it is worth noting that the effect is too weak to observe a Purcell-enhanced decay rate in the waveguide; the lifetime in the waveguide was fit as 11.4 ms, which agrees closely to the value reported in the literature, and is consistent with the 10.8 ms ms lifetime measured in bulk. The difference between fitted lifetimes in bulk and the waveguide can be accounted for by noise in the measurement, error in the fitting, and variation in the sample.[2] Before quantifying and discussing the enhancement of the decay rate for the ions coupled to the nanobeam and microring, we discuss the expected shape of the photoluminescence curve, determined by simulation.

[2]Note that the shorter, fitted lifetime of 10.8 ms in bulk listed in [70] results from fitting past the

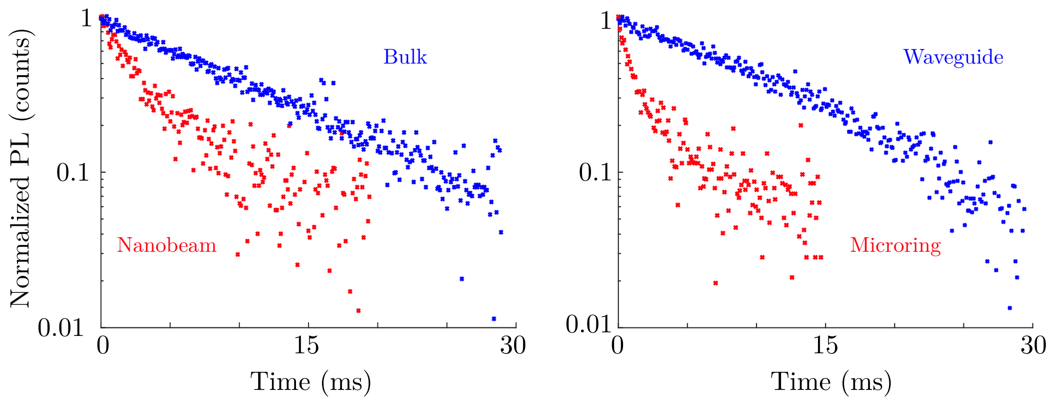


Figure 4.1: Purcell-enhanced decay in each of the cavities, with a comparison between the nanobeam and bulk PL, and between the ring and the waveguide, measured while the cavity was detuned. The lifetime of bulk and waveguide were fit by single exponentials with lifetimes 10.8 and 11.4 ms, which are consistent with measurements in the literature [32].

Simulating Photoluminescence

The emission of a photon for an excited ion is a Poisson process (continuous, independent, and with constant probability in time), and so the photoluminescence curve generated by histogramming the emission times should produce an exponential curve. However, different ions couple with different strength to the cavity, due to the inhomogeneity in the cavity field across the ions, included quantitatively as $\left(\frac{E(\vec{r})}{\max(E(\vec{r}))}\right)^2$ in Equations 1.4 and 1.11. This results in a distribution of Purcell enhanced decay rates for different ions. Using the simulated field profile of the cavity mode, we can calculate the value of $\left(\frac{E(\vec{r})}{\max(E(\vec{r}))}\right)^2$ as a function of location within the resonator. The nonuniformity in coupling across the ensemble is shown in Figure 4.2, which illustrates the fraction of ions as a function of electric field, electric field intensity (the aforementioned, squared quantity), and the electric field intensity weighted by the contribution to Ω at the location of the ion for each of the two resonators. Though it's not immediately clear from the log scale, due to the large number of ions weakly coupled to the ring resonator, the total cooperativity rates are very similar between the two cavities, though the total coupling rate for the beam is approximately three times higher.

Additionally, we can use the field profile to calculate the expected lifetime for ions at any simulated voxel in the resonator. Summing the contributions of the various ions, weighted by their individual probabilities of emitting into the cavity

initial, linear portion of the decay curve, which includes additional effects.

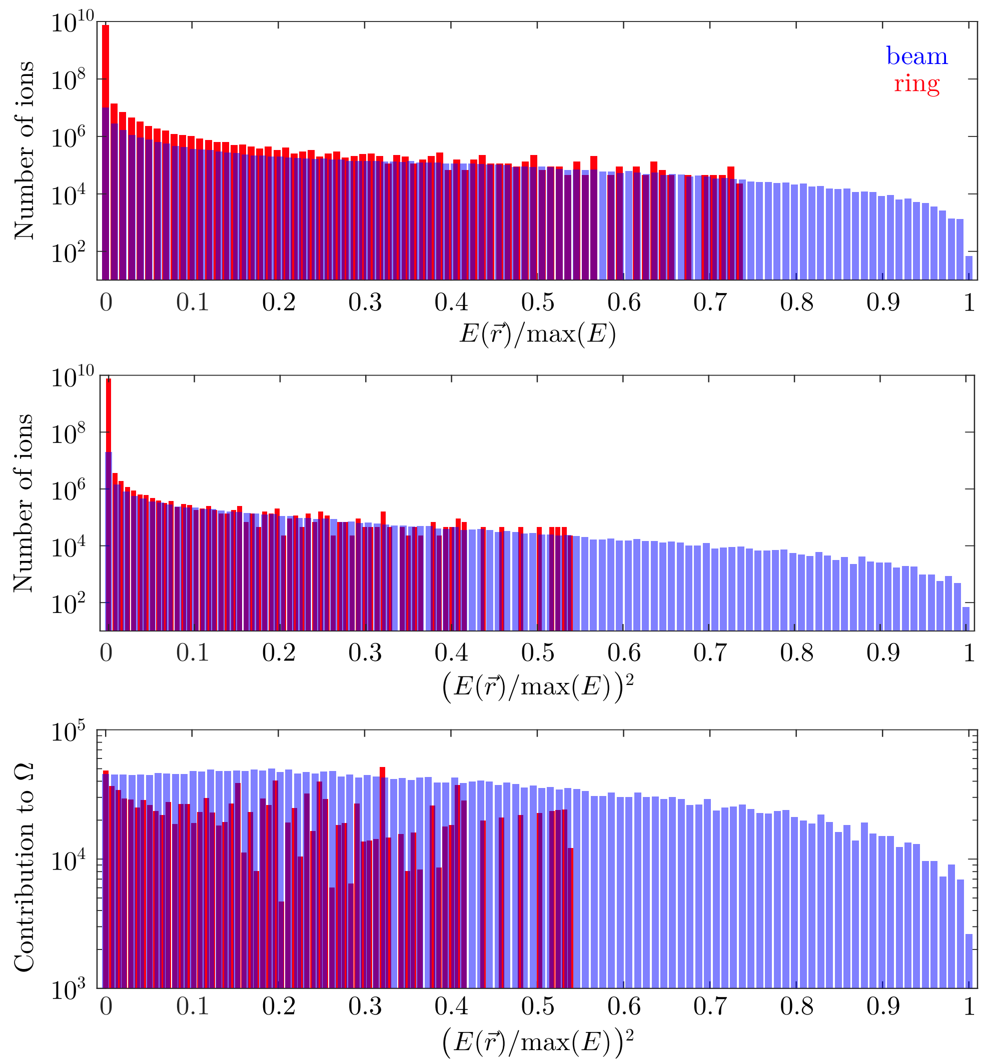


Figure 4.2: Histograms showing the number of ions as a function of coupling strength for each of the two resonators. The bin size is approximately 0.01, which corresponds to a 1% difference in $E(\vec{r})^2$. The granularity in the curves result from finite resolution in the simulation results. Note that the maximum value for the rings is approximately 0.75, while the nanobeam maximum value is 1, since the field maximum occurs in the YSO.

mode we can construct a predicted excitation decay rate curve for the ensemble as a whole, as a function of detuning between the cavity and the ensemble resonance. The expected photoluminescence intensity as a function of time for a continuous range of detunings can be computed from this, and is shown is Figure 4.3. Though the images appear similar, the scales of the ordinates differ by nearly an order of magnitude because of the factor of 10 increase between nanobeam and ring quality factors. Also, the higher total coupling rate Ω of the nanobeam results in faster decay overall, which manifests as a general shift of the PL decay to slightly shorter times.

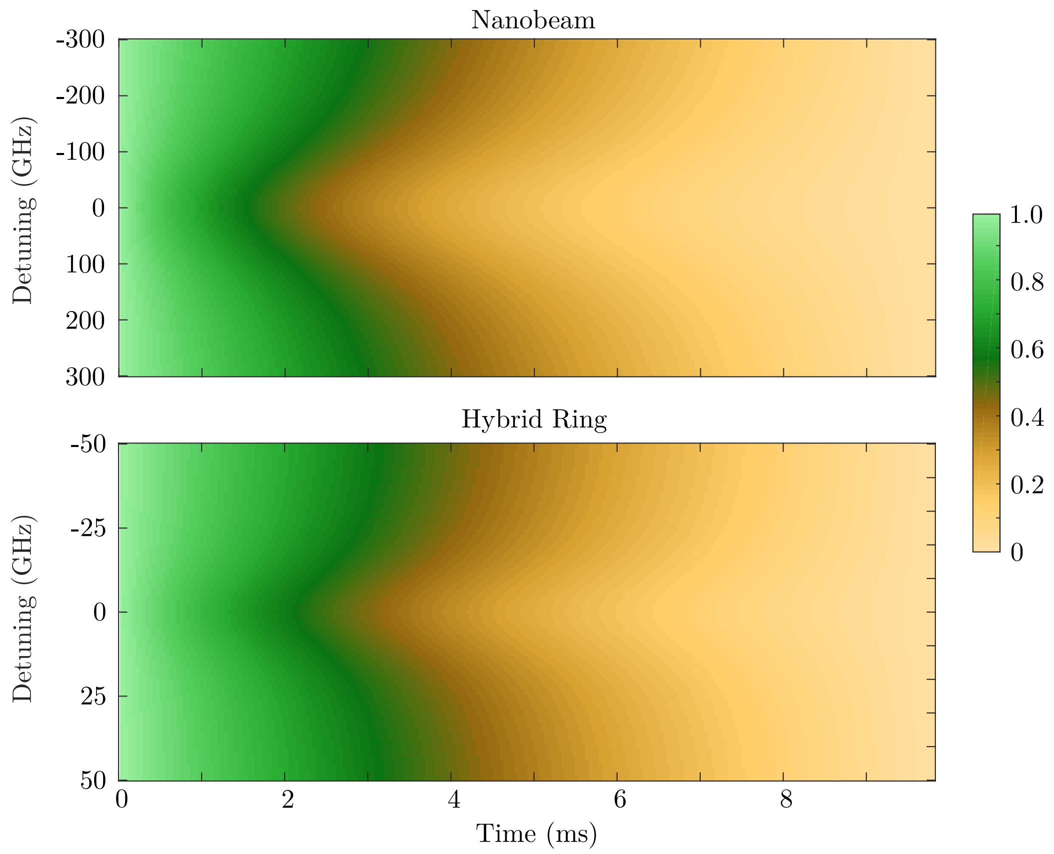
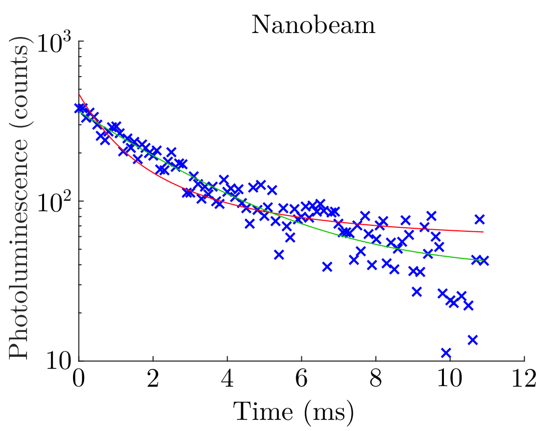


Figure 4.3: Simulated photoluminescence intensity as a function of time for various detunings. Note that the order of magnitude difference in cavity linewidth is reflected in the relevant scale of the detuning.

Fitting Photoluminescence

Alternatively, approximating the ensemble PL curve with a single exponential decay lifetime is useful when describing the modification of the system dynamics, like the improved spectral hole burning described in Section 1.3. To justify this, we compare and find good agreement between the measured PL from the cavity a single exponential fit to the data in Figure 4.4 for both the nanobeam and the ring resonator, where the single exponential was fit for lifetime, amplitude, and background. We also include a curve showing the simulated photoluminescence intensity, fit only for amplitude and background. The disagreement between simulated PL and the measured nanobeam data may result from the fabrication imperfections between the simulated and the measured device, as the simulated version perfect 60° and right angles, while angles of the fabricated version were limited by the nonzero focal spot of the gallium beam.



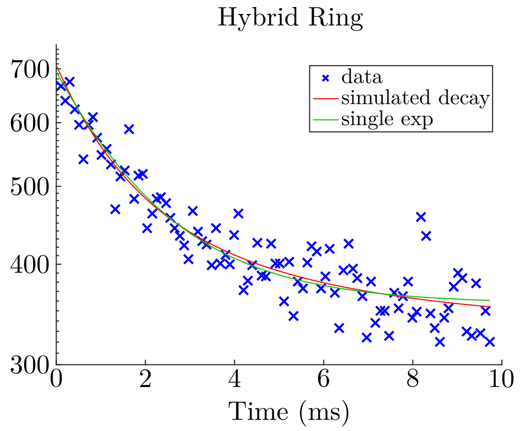


Figure 4.4: Photoluminescence intensity data for an ensemble coupled to a resonant cavity, compared to both the simulated PL curve fitted for amplitude and background, and a single exponential decay, fit for lifetime, amplitude, and background.

We aim to predict the lifetime generated from the single exponential fit by first logically extending Equation 2.3 for the total coupling rate, Ω, to an average of the coupling rate across an ensemble non-uniformly coupled to a cavity as

$$\frac{g_{\text{eff}}}{2\pi} = \sqrt{\frac{\int_{\text{YSO}} \left|E_z(\vec{r})\right|^2 \left(\frac{g_{\text{ion}}(\vec{r})}{2\pi}\right)^2 \rho \, dV}{\int_{\text{YSO}} \left|E_z(\vec{r})\right|^2 \ \rho \, dV}}, \tag{4.1}$$

where $\rho = 1.87 \times 10^6\ \mu\text{m}^{-3}$ is the ion density per unit volume for 200 ppm Er:YSO and the contribution of g_{ion} for each ion has been weighted by the proportion of the mode energy density at the ion, $\left|E_z(\vec{r})\right|^2$. For the ring resonator, we calculate $\frac{g_{\text{eff}}}{2\pi} = 0.211$ MHz, while for the beam, we find this value to be 1.33 MHz. Using this value and Equation 1.11, we can compute the average effective Purcell factor at zero detuning to be 6.26 for the ring, and 5.20 for the beam. Additionally, we can calculate the effective Purcell factor as a function of detuning, and compare to the lifetime for a single exponential fit of PL decay curves measured at various detunings. The single exponential lifetime prediction and fit as a function of detuning between cavity and ensemble is shown in Figure 4.5. The data points taken on for the nanobeam do not span the detuning curve enough to capture the gradual transition of the single exponential lifetime to reach that of bulk because the number of ions in the cavity is low enough that achieving a high signal-to-noise ratio on the photoluminescence is difficult without the cavity enhanced absorption and collection described in section

1.4. We also note that the lifetime of the transition over a detuning range far greater than the cavity linewidth. Quantitatively, the half width at half maximum of the lifetime vs. detuning curve is approximately $\kappa\sqrt{F_P\left(\frac{g_{\text{eff}}}{g_0}\right)^2+1}$.

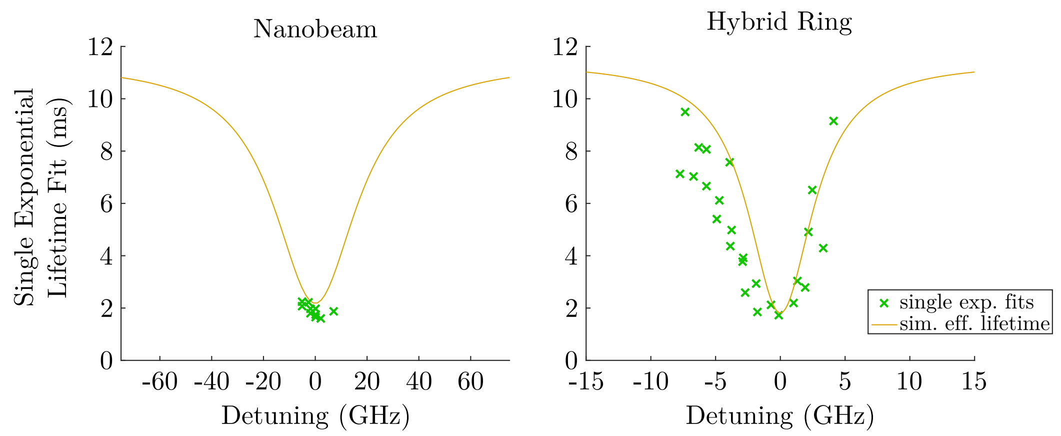


Figure 4.5: Effective Purcell factor as a function of detuning between the ion ensemble and the cavity. We compare the theoretical effective lifetime generated using measured cavity parameters and simulated mode profile with single exponential decay fittings from experimentally measured photoluminescence curves (in green both here and in Figure 4.4.

4.2 Comparison of Nanobeam and Hybrid Microring devices

While both devices accomplish the common goal of coupling an ensemble of erbium ions to a low-loss, microscale optical cavity, differences are numerous enough that a comparison and summary of relevant parameters is included below. Following this, we continue with a discussion of fabrication limitations on extending the devices, as well as comparing the efficiency and ease with which we can couple to the devices efficiently.

Devices Architecture Quantitative Summary and Comparison

A summary of the quantitative comparison of the two device architectures is presented in Table 4.1. Note that, though the effective Purcell factor and the cooperativity are relatively similar for the two architectures, this is due to an accumulation of happenstances and is not due to some fundamental limit set by coupling to an erbium ion ensemble in YSO. For example, new nanobeams fabricated more recently demonstrate quality factors which are higher by a factor of two, and should thus demonstrate a factor of two improvement in the cooperativity.

Because $g_{\text{tot}} \propto \sqrt{N/V_{\text{mode}}}$, the cooperativity, which scales as $g_{\text{tot}}^2/\kappa\Delta = Qg_{\text{tot}}^2/\Delta$,

Parameter	Nanobeam	Hybrid Ring
Highest measured quality factor	70,000	112,000
Mode volume (μm^3)	1.05	2.93
Maximum Purcell factor	517	285
Effective Purcell factor	7.0 (5.20)	6.6 (6.25)
Effective Coupling rate, g_{eff} (MHz)	0.211	1.33
Total coupling rate, Ω (GHz)	1.35 (1.60)	0.537
Cooperativity	0.48	0.54
One-way coupler efficiency	$\sim$ 20%	$\sim$ 2%

Table 4.1: Maximum values fitted from measurements and (simulated) for various parameters for the two optical resonator designs.Though the highest measured quality factor for the nanobeam is 25,000, all other parameters and measurements to this point were performed using an earlier beam with Q=11,400.

is dependent on the quality factor and the quantity (N/V_{mode}), which is very nearly the ion concentration ρ. As a result, a slightly higher cooperativity was achieved by the larger ring resonators, despite a lower g_{eff} due to the resonator's evanescent coupling, due to the increased quality factor, and because the large mode volume did not weaken the ensemble coupling rate. The coupler efficiency is included for because other concerns beyond individual device specifications must be considered to create scalable optical quantum memories.

Limits on Implementing Optical Quantum Memory Devices

An on-chip optical quantum memory would ideally be able to efficiently store large number of qubits with high fidelity for long storage times. The goal of storing many qubits requires scalability of fabrication. Additionally, temperature-dependent electron state dynamics require lower temperatures than those accessible in the continuous flow cryostat used in the above experiments, and engineering constraints place additional requirements on input and output coupler efficiency. These two concerns are discussed sequentially.

Fabrication methodologies continue to improve, but there are stark differences in scalability between the two devices. While far more susceptible to variations in manufacturing due to the manual nature of the FIB milling, the art and science of low-loss nanobeam fabrication has been steadily improving in the group, both in consistency between beams as well as quality factor of completed devices. Recent devices fabricated by Jake Rochman show quality factors up to 70,000 in YSO, and quality factors of up to 50,000 have been achieved in YVO. These high loaded quality factors were achieved by improving fabrication processes, further optimizing the design, and decreasing the input coupling rate κ_c. While these measurements place a lower bound on the intrinsic quality factor, Equation 1.8 shows that κ_c must be equal to the absorption rate of the ions to maximize the echo efficiency, and both must be much greater than κ_i; this means the loaded Q must be intentionally degraded by increasing κ_c. After doing so, it becomes difficult to determine Q_i easily from measurements of the much smaller Q_l and, because of the serial nature of the fabrication, it is difficult to guarantee consistent values of Q_i across multiple devices.

Additionally, there is an inherent difficulty in accurately milling structures that are more than about twice the length of the nanobeam resonators. To do so entails either aligning features larger than the scan-size of the FIB system at high magnification, or the accurate fabrication of devices at low magnification. As a result, fabricating waveguides to couple milled cavities to on-chip sources, detectors, or each other is daunting. By comparison, electron beam lithography and dry etching used in the fabrication of the hybrid resonators allows trivial extension of the demonstrated fabrication process to generate chip-scale waveguides coupling an arbitrary number of resonators, with straightforward extension to include on-chip detection. However, implementing on-chip sources is formidable, and thus efficient coupling of off-chip sources to the devices is paramount.

As stated in Section 1.3, spin state lifetimes in Er:YSO are only around a factor of 10 longer than the lifetime of the optical exited state (the lowest $^4\mathrm{I}_{13/2}$ state), depending on magnetic field strength and orientation, which results in poor state initialization. Cooling below ≈ 3 K can extend the spin lifetime to the point where population can accumulate in auxiliary ground states [49], which involved the use of a helium-3 refrigerator. Engineering constraints, which will be described in the following section, required that the device be measured in reflection from free space. As a result, the efficiency of the optical coupling is an important parameter to compare between the two devices.

The efficiency of the nanobeams' trench-end coupler was measured to be approximately 20% [92], and was high enough that coupling to the cavity in reflection using a single port could be performed with sufficient signal to noise. The largest contribution to the noise was the Fresnel reflection of 8% of the incident power off of the top surface of the YSO, which was lower than the 4% round-trip coupling efficiency because the Fresnel reflection inefficiently coupled back into the single mode input fiber. The efficiency of the rings' grating couplers, however, was measured to be a much lower value, at 0.04-0.05% round trip, which comes out to $\approx 2\%$ for each grating. Because of the low coupling efficiency, measurement of the hybrid devices required two different grating couplers to collect transmission through the ring. This could be remedied using a substantially more complex fabrication procedure to create a mode-matched waveguide for coupling with a lensed fiber or a self-aligning fiber coupler [93, 94], where single-pass efficiencies of 34.7% [95] and 86% [96] have been demonstrated in experimental systems. However, low efficiency at present prohibited single port measurement, which precluded easily measuring the devices in the ^{3}He fridge with a single stack of nanopositioners. As a result, all further measurements were performed on triangular nanobeam devices.

4.3 Beyond a Two Level System

Temperatures lower than the 4 K boiling point of liquid helium, required for extending spin lifetimes to experimentally relevant timescales, were accessed using a BlueFors BF-LD250 system, which used a pure helium-3 cycle on the "still plate" to reach ~500 mK.[3] The system is a dry fridge, using pulse tubes to cool a still plate where helium-3 is compressed and condensed, and gaseous helium-3 is pumped out to be recompressed again. The still plate is mounted to and in weak thermal contact with a plate at ≈ 4 K, which is similarly mounted to another plate at ≈ 45 K, which is mounted to the top plate of the system at room temperature. The underside of each plate is enclosed by a radiation shield, and all cooled components in the system are enclosed in a vacuum chamber consisting of a cylinder attached to the top plate.

Device Coupling in the Helium-3 Fridge

The samples with milled nanobeam devices were indium-soldered to a custom-machined sample mount, screwed onto an xyz Attocube stack (ANP101/RES) which was mounted on a custom machined U-shaped component screwed into the underside of the still plate. A copper braid provided thermal connectivity between the sample mount and the still plate. Photographs of the sample mount and optics are shown in Figure 4.6. All custom components were machined from oxygen-free high thermal conductivity (OFHC) copper and most were gold-plated to improve thermal conductivity by optimizing hot electron transport across the interfaces.

Optical access to the vacuum chamber in the telecom C band was achieved using

[3]The unit was purchased without a mixing chamber or the turbo pump required to implement a full dilution refrigerator, but with the capability to add the necessary components in the future.

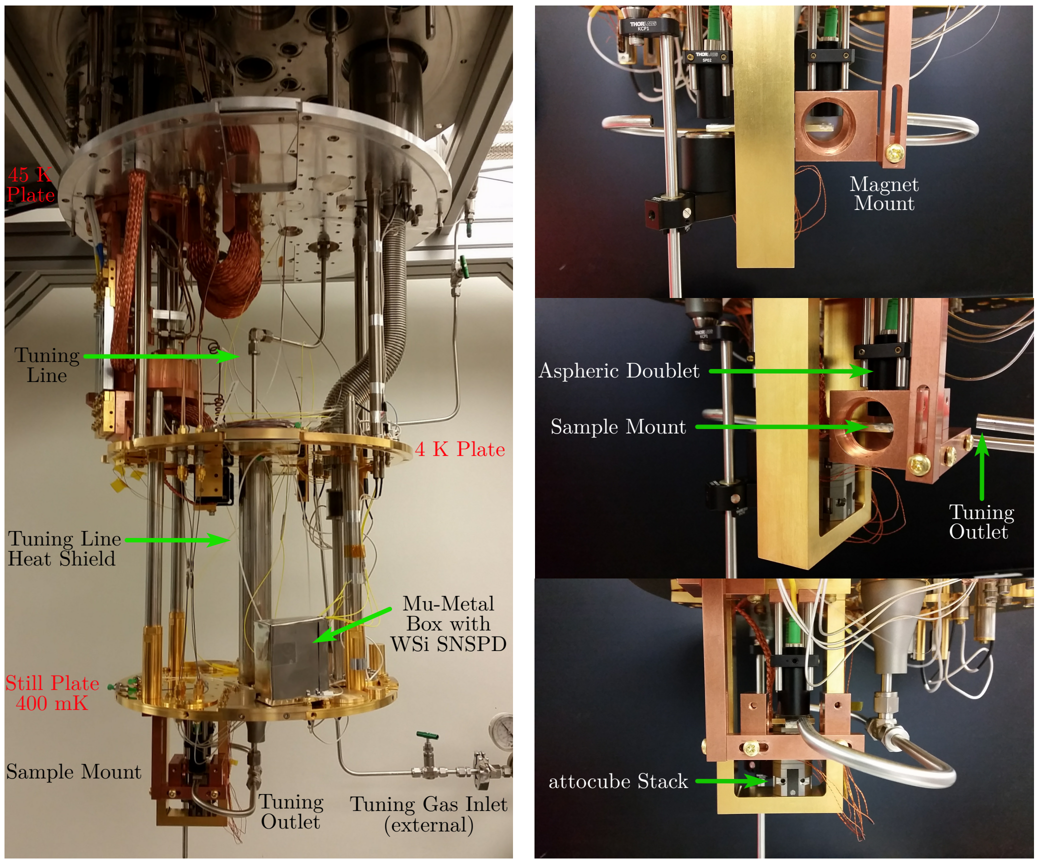


Figure 4.6: Sample mount and optics for coupling to devices withing the Bluefors fridge. The 45 K, 4 K, and 400 mK plates are labeled in red. The mu-metal box housing the SNSPD and the 1/4" stainless steel tube used for gas-condensation tuning are also indicated. On the right, three images show different angles of the sample mount on top of the attocube stack, sitting on the 'U' mount attached to the bottom of the 400 mK plate. The magnet on the near side has been removed from the mount to allow viewing of the sample through the mount.

single-mode optical fibers (SMF-28e). The fibers were FC/APC connectorized on one end and plugged into mounted fiber connectors on the front panel of a light-tight break-out box. The other end of the fiber was guided out of the box, through vacuum bellows and a hole in a KF-40 flange; the latter was sealed with epoxy (Loctite STYCAST 2850FT) and the flange was attached to a vacuum feedthrough on the top plate of the fridge. Though the fiber is at atmospheric pressure within the box and bellows, vacuum hardware is used to provide a light-tight environment. The break-out box and epoxy-sealed feed-through are shown in Figure 4.7.

Inside the fridge, a bare fiber end was cleaved and spliced to an FC/APC connectorized fiber which was plugged into a free-space coupler and aligned to an

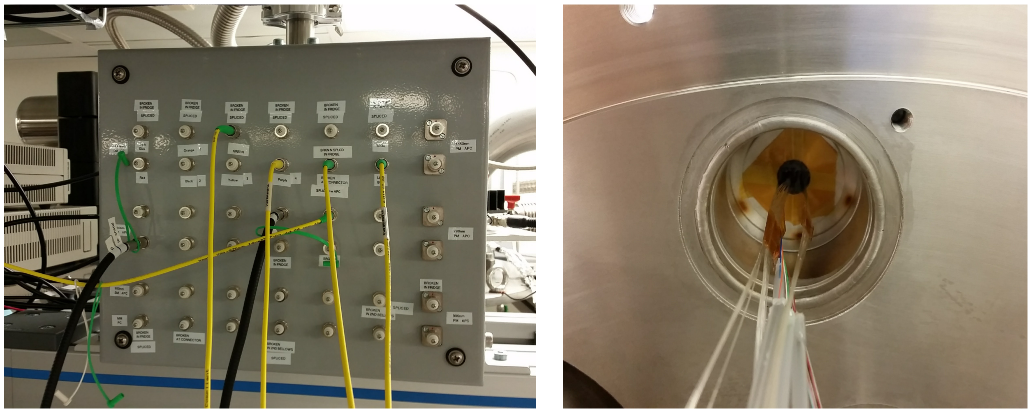


Figure 4.7: Optical access to the Bluefors fridge is achieved using optical fiber. The image on the left shows the breakout box with connectorized access to 42 fibers, including single mode and multimode fibers with transparency windows including 880 nm, 980 nm, and 1550 nm. The fibers leave the box through the tube in the top, through a bellows and through a drilled hole in a blank KF-40 flange. The KF flange and epoxy-sealed hole are shown on the right, viewed from the inside of the vacuum chamber.

aspheric doublet, both of which were mounted in a commercially available 16 mm cage optomechanical system. The posts from this cage system were mounted into custom machined mounting components, mounted to the still plate of the fridge. On top of the still plate, another of the fibers passing through the epoxy-sealed vacuum feed-through was similarly spliced and connected to a self-aligning WSi nanowire taper SNSPD inside a mu-metal box, which can also be seen in Figure 4.6. The detectors were fabricated by Matt Shaw and his group at JPL [90].

Optical coupling between the fiber and the devices was accomplished performing coarse alignment when mounting and fine alignment using the attocube stack. Spacing between the sample and the devices was initially set adjusting the cage mount, before using the attocube z positioner to maximize the reflected signal. Prior to mounting the sample, the position of the devices was determined relative to a corner of the sample using the confocal microscope. A map of the sample was then generated using the attocube x and y positioners. Scanning was typically performed at or above 4 K, before running the helium-3 condensation cycle, as attocube responsivity and repeatability was low at ~ 400 mK. An example image of the beams generated by this scan is shown in Figure 4.8. Finally, coupling was achieved and optimized by manually rastering the positioners in the vicinity of the coupling trench while observing the collected reflection on the spectrometer and watching for the characteristic spectral feature of the cavity.

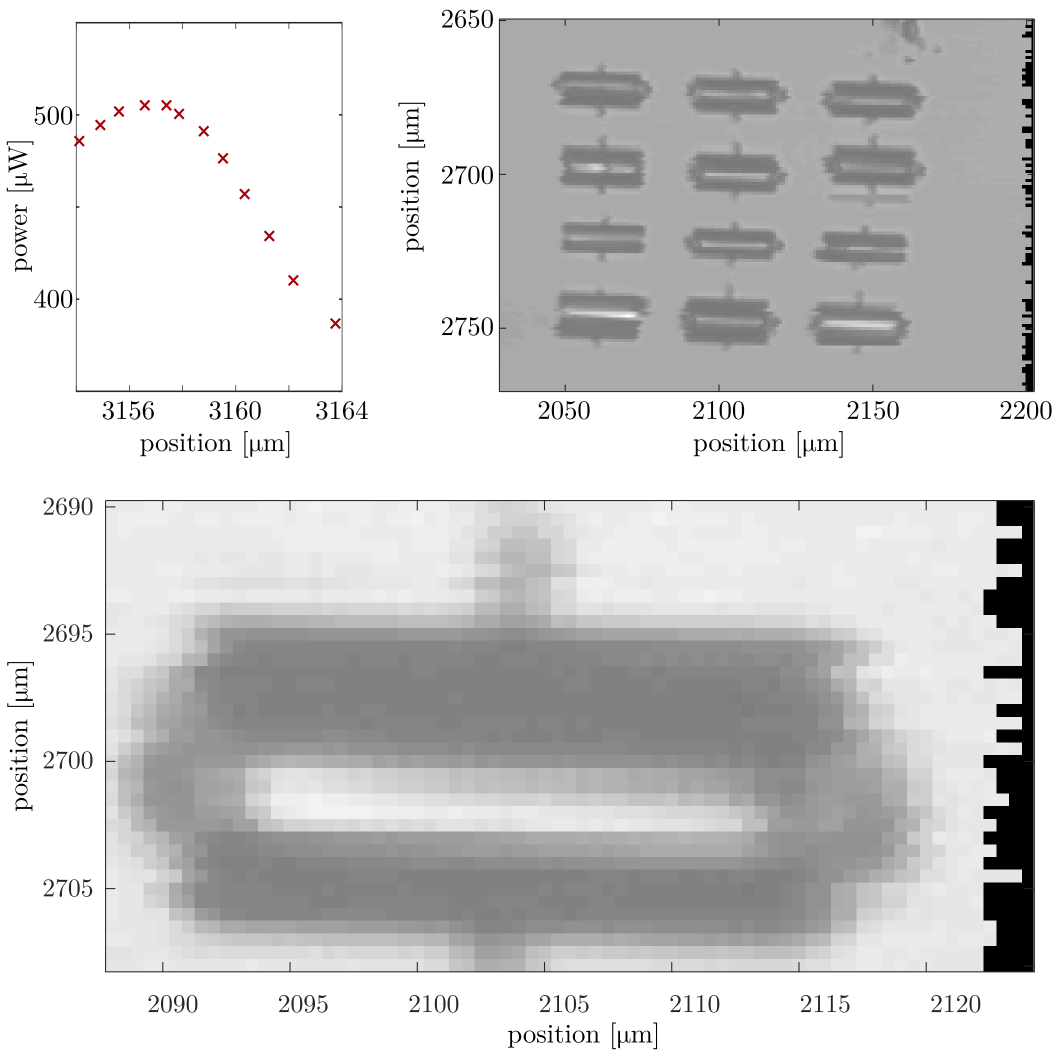


Figure 4.8: The reflected power as a function of the z position of the attocube is shown on the left, with a clear maximum indicating the light is focused on the top surface. A coarse scan of 12 devices is shown on the right, and a fine scan is shown on the bottom. The black artifacts on the right side of the two scans results from inconsistent distanced traversed by the x attocube in the given number of steps.

The optical system used to characterize the sample is shown in Figure 4.9. Samples were optical characterized using only fiber components, and a fiber circulator was used to separate input light from output light. The input light was amplitude modulated by an AOM and frequency modulated by the laser, and the output was sent to either the spectrometer, fast photodetector, or back into the fridge to the WSi SNSPD by manually disconnecting and reconnecting fibers. Additionally, the output could be amplitude modulated using an EOM (or later, a second AOM) to avoid sending too much light to the SNSPD. The apparatus allows light to be coupled to and from the devices, analogous to the capability of the confocal microscope, except exclusively one optical mode is measured, limited by the single mode fiber.

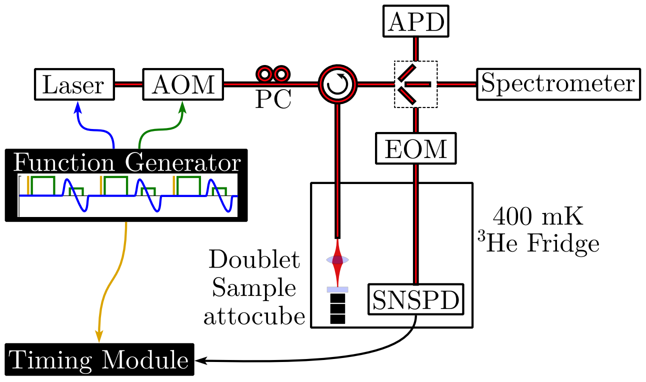


Figure 4.9: The fiber optic system used to characterize the devices in the fridge at 400 mK, illustrating input frequency control with the laser's internal piezo control, amplitude control with the AOM, and polarization control (PC), as well as the various detectors which can be selected by disconnected and reconnecting fiber.

Additionally, mounts were machined and mounted to the still plate to hold two 1" diameter cylindrical magnets on either side of the sample to apply a static magnetic field. By choosing different sized magnets and adjusting the spacing of the magnets, the applied field could be changed continuously from 5 to 120 mT while the fridge was vented. The magnet mounts and magnets are also visible in Figure 4.6.

Cavity-Enhanced Absorption Tailoring

With this system constructed, we were able to burn and probe spectral holes in the devices in Er:YSO using the lowest doublets of the $^4I_{15/2}$ and $^4I_{13/2}$ manifolds. As described in Section 1.2 and 1.3, the lifetime and branching ratio of the optical excited state depend on magnetic field strength and orientation. For measurements on this beam, we used a magnetic field of 60 mT applied at angle of 120° from $\mathbf{D}_1$ toward $\mathbf{D}_2$. According to measurements in the literature [97, 47], an angle of approximately 100° or 135° will maximize the hole burning depth, likely differing due to the strength of the applied field and erbium concentration.

To allow gas tuning of the nanobeam cavities inside the fridge, a 1/4" stainless steel tube conveyed nitrogen gas from the top, room temperature plate of the fridge down to the sample. The tube was thermally and mechanically connected at the 45K plate of the fridge, and surrounded by a stainless steel tube thermally and mechanically mounted to the 4 K plate, acting has a heat shield. The tuning line itself can be seen in Figure 4.6. A tuning curve illustrating the resonance of the device moving across the two spin-preserving transitions (ω_2 and ω_3 in Figure 1.6) can be seen in Figure 4.10; the two cross-transitions are too weak to be visible in the plot, unlike in the schematic illustration, Figure 1.6. The spin-preserving transitions appear to change from a peak to a dip as the cavity is tuned, due to a predicted Fano interference between the light in the cavity and the ensemble described well by Equation 2.1 and 2.2 from [80].

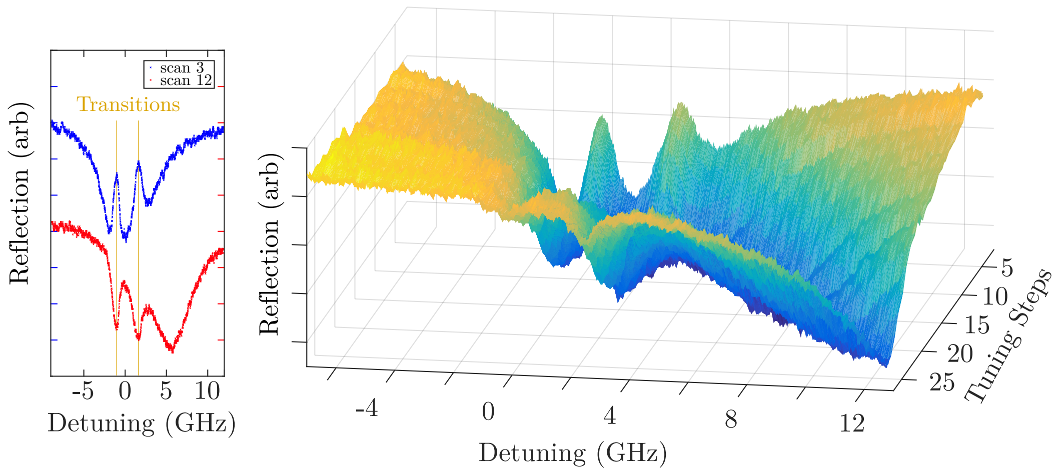


Figure 4.10: Reflection amplitude off the trench-end coupler of the resonator as the resonator is tuned through the optical transition of the erbium ions. The plot on the left shows two scans of the cavity illustrating the two narrow spectral features corresponding to the spin-preserving transitions of the erbium. Scans are separated vertically for clarity. A surface constructed of scans along multiple tuning steps is shown on the right. The transitions appears to change from a dip in the foreground to a peak on the far side as the cavity resonance passes due to an interference effect.

Spectral hole burning was demonstrated in the cavity-coupled ensemble by leaving the laser at one frequency, then reducing the intensity and scanning across the transition while recording the intensity. Due to the small size of the ensemble and fairly high cooperativity, it was overly arduous to obtain a scan of the full ensemble without saturating the transition or having a signal indistinguishable from noise. Locally, the hole could be observed, but a baseline for absorption, and hence accurate measurement of the hole depth and the initialization efficiency could not be measured.

By burning at multiple frequencies, population can be driven from different points in the inhomogeneous line to create a spectral structure in the ensemble. The experimental parameters were varied to optimize hole depth for this two-toothed comb: 75 repetitions of 500 μs burn pulses at three separate frequencies approximately 14 MHz apart, with 500 μs between burn pulses to tune the laser were used to define the teeth. The resulting atomic frequency comb in the nanobeam cavity is shown in Figure 4.11. Unfortunately, the aforementioned saturation problem precluded an accurate determination of the comb contrast. The deepest spectral holes that we could burn in bulk yielded holes that reduced the total optical depth by 5%. As the hole burning should be enhanced by the cavity, we can use the bulk efficiency as a conservative estimate for the hole burning efficiency measured in bulk. If we additionally assume that 30% of the cavity loss is coupled to the input channel, Equation 1.8 predicts that the echo efficiency from this AFC would be less than 0.1 % efficient, which was below the noise floor of the measurement. The dominant term handicapping the efficiency was the absorbing background due to poor hole burning. However, a promising method of improving hole burning efficiency and echo efficiency will be discussed in the remainder of the thesis.

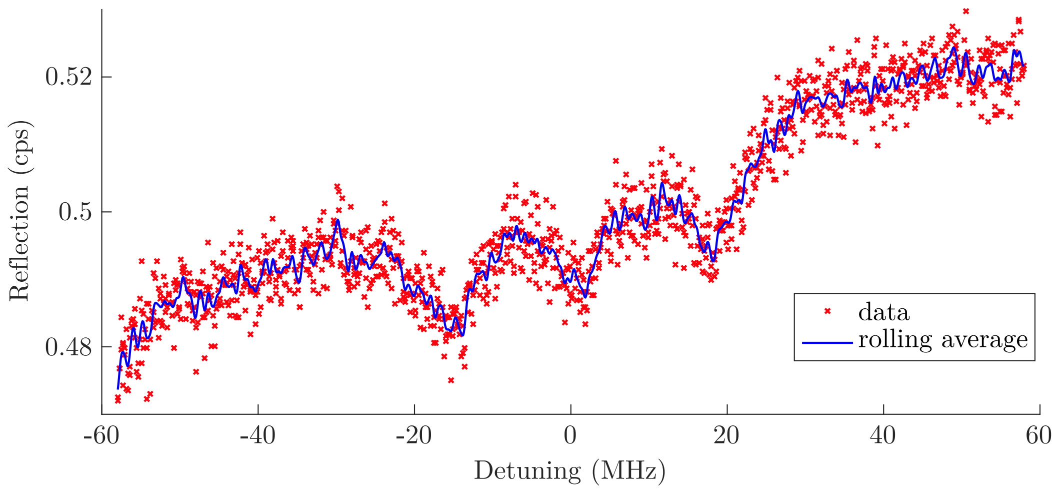


Figure 4.11: Reflection scan of an atomic frequency comb burned into the ground state of the ensemble inside the nanobeam cavity. The blue line illustrates a rolling average of 7 points. Due to low count rates, this reflection spectrum required an integration time of 8 hours.

4.4 Photon Echoes

Additionally, we were also able to probe the coherence of the optical transition for the erbium ensemble in the nanobeam using two pulse photon echoes. Figure 4.12 shows three echoes, with pulse delays of 440 ns, 1.3 μs, and 3 μs. The high intensity of the $\pi/2$ and π pulses required for the echo resulted in latching of the detectors. From the shape of the coherence decay curve, also shown in

Figure 4.12, we can extrapolate a decay constant of 800 ns. Because the pulse spacing is half the dephasing time, and the echo height is proportional to amplitude while T_2 describes the decay of the wavefunction (and is thus a field, not intensity parameter), this decay constant corresponds to a T_2 of 3.2 μs. The oscillation on the exponential decay of the lifetime curve is believed to be the result of superhyperfine interaction between the net electron spin of the erbium ions and the nuclear spin of the yttrium ions, since the period of the sinusoidal oscillation has a period of approximately 0.6 MHz, which is fairly consistent the strength of the superhyperfine interaction predicted and measured in the literature [98]; any discrepancy can likely be accounted for error arising from measuring only approximately one oscillation of the sinusoidal variation.

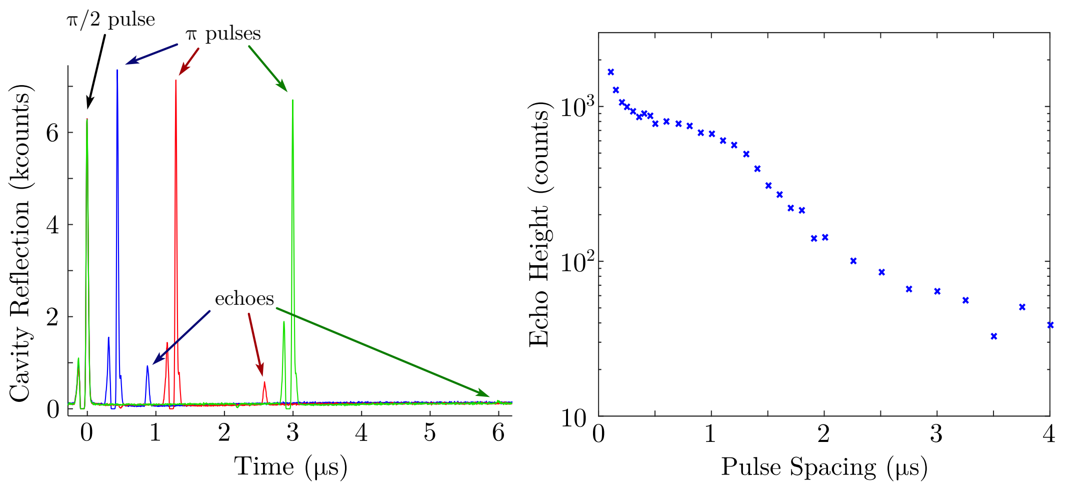


Figure 4.12: Measured echo intensity measured from the triangular nanobeam in reflection. The distortion of the $\pi/2$ and π pulses is the result of saturation.

For the nanobeams in Nd:YSO, the coherence time was measured in both the nanobeam and bulk, and it was determined that the coherence time was not degraded by the fabrication process itself. We were unable to measure photon echoes when reflecting off the back side of the bulk Er:YSO sample due to high absorption; the 500 micron thickness exhibited an optical density of approximately 5, and due to k-vector matching, the echo, reemitted in the forward direction, would be strongly attenuated by ions not excited by the driving pulses. However, the coherence time measured in the beam is consistent with the measured values presented in the literature [33] for the same concentration at higher temperatures and higher magnetic fields.

Chapter 5

ERBIUM-167 AND FUTURE DIRECTIONS

In this chapter, I discuss very recent progress and future directions. I discuss the prospects of using isotopically pure ^{167}Er:YSO for telecom C band quantum memories. Specifically, we motivate the use of the hyperfine levels of erbium-167 in high magnetic fields, present the results of bulk hole burning and atomic frequency comb preparation, and present future directions combining the previous devices with hyperfine structure of erbium-167.

5.1 Hyperfine Structure

In order to implement long-term spin-wave storage using an AFC protocol, one optical excited state and three metastable ground states are required. As discussed in Section 1.3, the population is moved via spectral hole burning from the ground state of the optical transition to a storage state; after the write pulse is absorbed by the comb, the resulting excitation on the optical transition is shifted to an excitation between the ground state and an auxiliary ground state using light at a different frequency. While using the hyperfine levels of erbium-167, the only erbium isotope with nonzero nuclear spin, has always been a long-term consideration, implementation of any protocol on the levels is difficult due to the large number of ground and excited hyperfine levels.

For low and moderate applied magnetic fields, the 16 ground and excited states exhibit 256 optical transitions that are unresolvable when scanning the inhomogeneous line. Nonetheless, work has been done to identify lambda-like systems (where two metastable ground states share an excited state) in ^{167}Er:YSO that were used to demonstrate electromagnetically induced transparency [52]. Additionally, microwave resonators have been used to directly excite the hyperfine transitions in ^{167}Er:YSO to more accurately determine the spin Hamiltonian parameters [99]. Lastly, and perhaps most importantly to the task of producing scalable telecom optical quantum memories, one effort probed a bulk ^{167}Er:YSO sample under very high magnetic fields, separating spin flip levels to be much larger than the peak in the phonon occupation spectrum, which provided three important results [41]. Firstly, it was found that optical pumping of the ensemble could achieve very long-lived ($\sim$10 minute decay constant) and highly efficient state initialization. Secondly, initialization of the population in a single hyperfine ground state reduced the number of active transitions and allowed the remaining transitions to be spectrally resolved. Lastly, the coherence time between hyperfine transitions was shown to be greater than a second at 1.4 K. For comparison, one second is long enough for signals in an optical fiber to circumnavigate the globe over five times, and thus sufficiently long storage time to implement quantum repeater protocols for a global optical network.

However, our experimental apparatus is, as yet, unable to reach magnetic fields of this magnitude. As hyperfine coherence time was found to possess a local maximum at lower magnetic field, until reaching a field of over 2 T, yielding a splitting of approximately 500 GHz, our initial exploratory work included here was performed with a 60 mT field applied along the $\mathbf{D}_2$ axis. This work includes a demonstration of hole burning with substantially higher state initialization efficiency than the erbium dopant ensembles with natural isotopic abundance, as well as the burning of atomic frequency combs and the measurement of atomic frequency comb echoes.

5.2 Hyperfine Level Hole Burning

A bulk sample of ^{167}Er:YSO was indium-soldered onto the attocube sample mount shown in Figure 4.6 and characterized using the fiber optic system illustrated in Figure 4.9. The sample, obtained from Scientific Materials, was doped with 50 ppm ^{167}Er and measured 1.0 mm x 1.5 mm x 1.5 mm in the $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{b}$ directions, respectively. Light coupled from the fiber and aspheric doublet above the sample mount propagated along the $\mathbf{D}_1$ axis and was focused on the back side of the sample. The incident light was polarized parallel to the $\mathbf{D}_2$ axis using polarization paddles to maximize absorption.

A location in the inhomogeneous line was identified which exhibited particularly efficient hole burning. A set of three holes were burned stepping between the three burning frequencies, waiting 100 ms, and then scanning the inhomogeneous absorption line. The holes are shown in Figure 5.1, and reach a depth of 0.2 OD from a maximum of 0.6. From this, we can conclude that two thirds of the equilibrium population has been moved to and remains in other hyperfine states.

5.3 Atomic Frequency Combs

Two methods were implemented in our attempts to create atomic frequency combs and measure AFC echoes. The first method involves using pairs of short pulses generate a comb in frequency space. The second method entails alternating between burning trenches to define a tooth and manually tuning the laser between

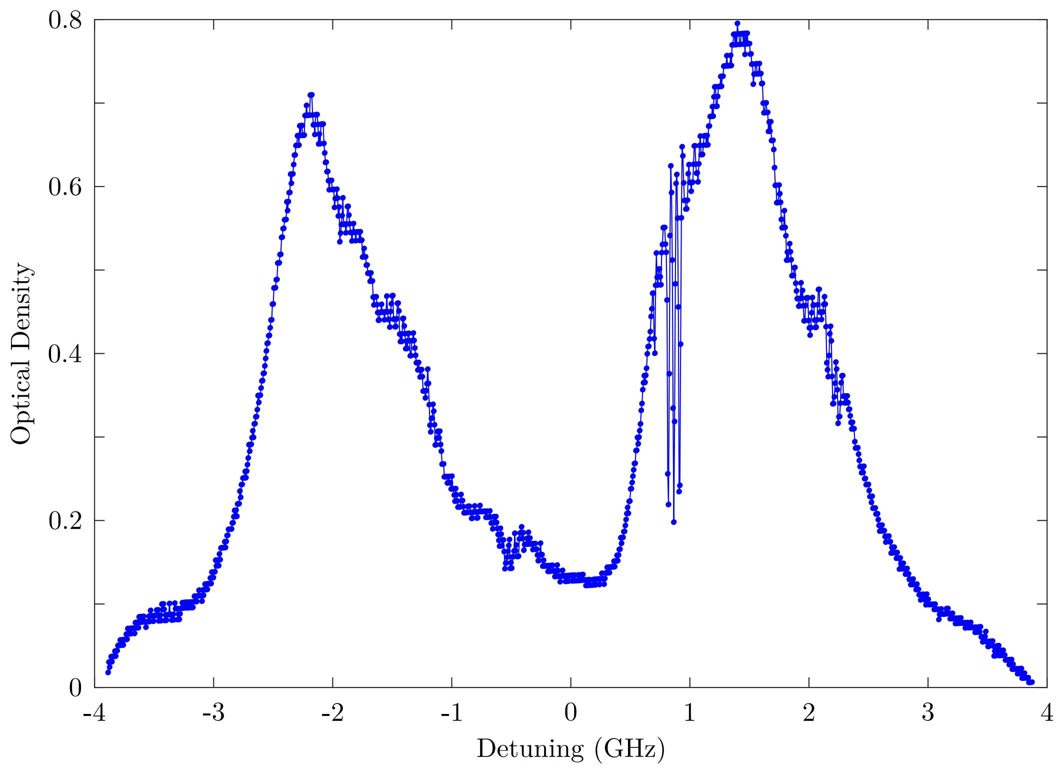


Figure 5.1: Absorption measurement of the inhomogeneous spectrum of bulk ^{167}Er:YSO with three spectral holes. Numerous anti-holes and side-holes can be identified by the presence of three peaks or dips. The actual data points are shown to emphasize the low density of points, while the line between points aids in identifying the narrow spectral features.

stops. Greater success was achieved using the latter method, but progress implementing both procedures is presented.

Accumulated AFCs

Accumulated atomic frequency combs are formed using pairs of laser pulses to excite ions according to a sinusoidal distribution in frequency. This relation follows directly from the Fourier transform of two pulse pairs, namely

$$\begin{aligned} \text{Intensity}(\omega) &\propto \mathscr{F}\left\{\text{rect}(t/a) * (\delta(t-t_0)+\delta(t+t_0)\right\} && (5.1)\\ &= \mathscr{F}\left\{\text{rect}(t/a)\right\} \cdot \mathscr{F}\left\{(\delta(t-t_0)+\delta(t+t_0)\right\} && (5.2)\\ &= \left(a\sqrt{\frac{2}{\pi}}\right)\text{sinc}(a\omega/2)\cdot\cos(\omega t_0), && (5.3)\end{aligned}$$

which is precisely a sinusoidal intensity variation on a sinc envelope. It's worth noting that the period depends on the spacing between the pulses and the width of the sinc is inversely proportional to the width of the individual pulses.

When burning accumulated combs, it is possible to use multiple pairs of pulses, in order to excite more ions and deepen the comb contrast, provided the spacing between pulse pairs exceeds the coherence time of the optical transition. In this manner, 1000 pairs of 40 ns pulses were used with 100 ns between pulses to generate the first comb depicted in Figure 5.2. The second comb was generated using 4000 pairs of 20 ns pulses spaced by 220 ns.

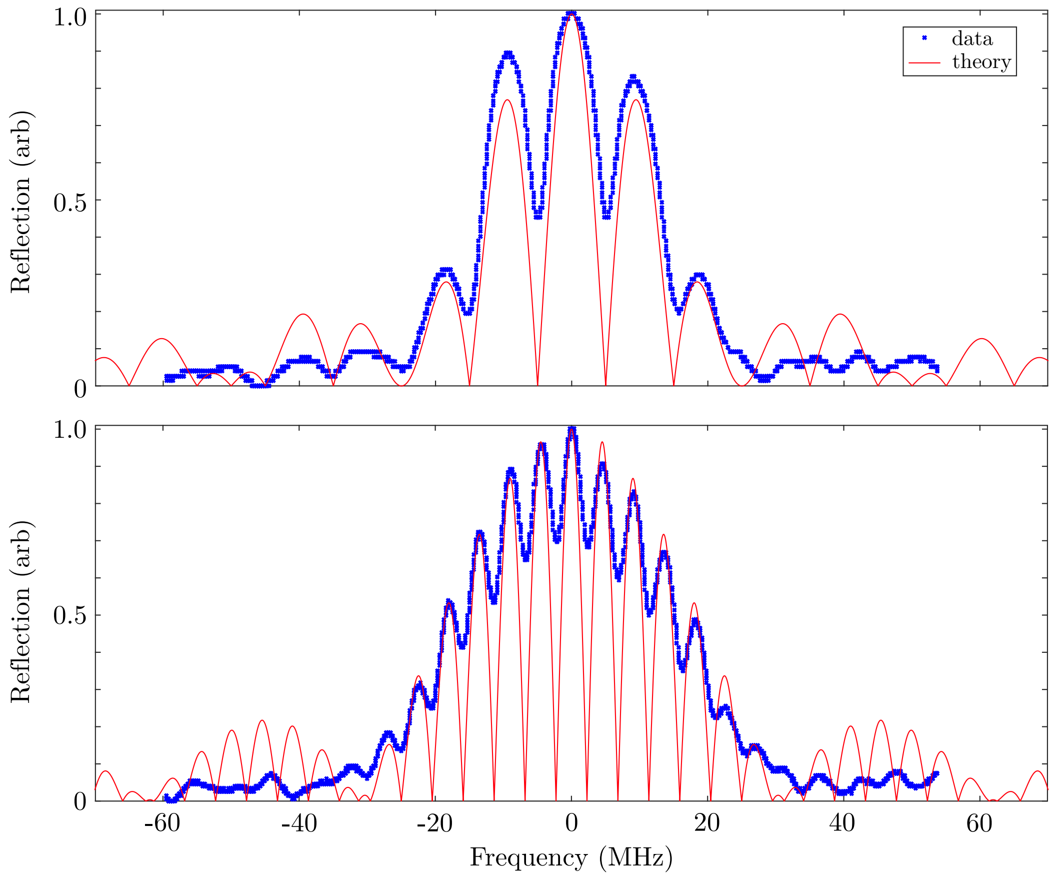


Figure 5.2: Measured and simulated atomic frequency combs in ^{167}Er:YSO, generated using a train of pulse pairs. The upper comb was generated with 40 ns pulses 100 ns apart, while the lower comb was generated with 20 ns pulses spaced 220 ns apart. The lower comb was best fit assuming 32 ns pulses.

There's a noticeable asymmetry across the two sides of the comb, arising solely as an artifact of the scan direction. In addition to the data, we show the predicted curve generated by computing the Fourier transform of the input pulses. Because the rise time required to generate 20 ns pulses is past the bandwidth limit of the AOM used to generate the pulses, the lower comb is better modeled by 32 ns pulses. Additionally the envelope of the comb deviates from the sinc due to the imperfect rectangular shape of the pulses themselves.

A large comb bandwidth is ideal; however, a minimum of two teeth are required to obtain an echo. Additionally, there is a limit on the width of an individual AFC comb tooth, resulting from the homogeneous linewidth and spectral diffusion of the erbium ions. Unfortunately, due to the characteristic size of holes that can be burned, even this weak constraint of two teeth creates a requirement on the width of the individual pulses that can be used when driving the optical transition. These narrow pulses contain a very low optical power, and as a result, many pulse pairs must be used. However, given requirements on the separation of pairs, increasing the number of pulses approaches an asymptotic limit and does not yield arbitrarily efficient hole burning. The inefficient hole burning is apparent when observing the absorbing background between the teeth of the comb. Due to this background, and the variation in tooth height across the comb, we were unable to measure AFC echoes using this comb.

Piecewise-defined AFCs and Echoes

More success was achieved in burning piecewise-defined combs, as shown in Figure 5.3. To generate this feature, 2 ms burn pulses were spaced by 500 μs, during which the laser was tuned 5.6 MHz using the internal piezo-mounted grating. The pattern was reinforced 25 times with 500 μs between repetitions and a 10 ms wait before scanning the comb. We observe that the maximum optical depth of the comb is $d = 0.38$ OD, the absorbing background is approximately $d_0 = 0.18$ OD, and the finesse is approximately $F = 2$. The efficiency of an AFC echo is given by

$$\eta = \left(\frac{d}{F}\right)^2 \exp\left[-\frac{d}{F} - \frac{7}{F^2} - d_0\right], \tag{5.4}$$

derived by Afzelius, et al. in [59]. Using this, we can calculate the expected efficiency for an AFC echo from this comb to be 0.4%.

Using this comb, we were also able to measure AFC echoes in bulk ^{167}Er:YSO, shown in Figure 5.4. For comparison, the echo is also shown from a comb with teeth spaced 5.6 MHz, using the same pulse parameters. The measured efficiencies for the shown echoes are estimated to be 0.17% and 0.22% for the 5.6 MHz and 6.6 MHz combs, respectively. The factor of two difference between expected and measured efficiency likely results from difficulty estimating the optical density of the comb itself.

The fabrication of a cavity coupled to these erbium-167 ions should dramatically improve both the preparation of the comb and the efficiency of the echo. To predict

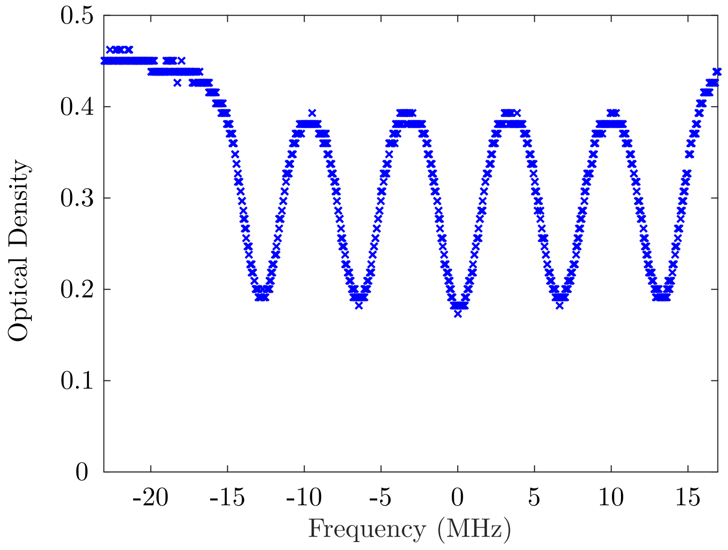


Figure 5.3: Measured atomic frequency comb in ^{167}Er:YSO generated by burning 5 holes sequentially to create each of the four teeth. The comb teeth spacing is 6.6 MHz. The optical density axis appears to possess discrete values due to the limited resolution of the oscilloscope digitization of the linear APD output.

the efficiency of future devices, we assume $(\kappa_c/\kappa) = 0.5$ and a quality factor of 20,000, which corresponds to an intrinsic quality factor of 40,000.[1] Using Equation 1.8 from [100] and neglecting improved comb initialization from the cavity, we can compute the expected efficiency to be 0.5%, which already more than doubles the bulk echo efficiency. In a realistically optimistic scenario, we first assume that we can reproduce the 95% initialization efficiency in the high magnetic field regime [41]. Then, if we assume a finesse of 5 and $(\kappa_c/\kappa) = 0.75$, which requires that the loaded quality factor is one quarter of the intrinsic quality factor and use the highest measured nanobeam quality factor of 70,000, this gives us a loaded quality factor of 17,500. Substituting these values into Equation 1.8 gives a predicted efficiency of 17.6 %. While these are optimistic values, all are based on measured parameters or existing devices. Furthermore, the Purcell effect would increase the ratio of excited state lifetime to hyperfine lifetime, improving hole burning efficiency for cavity-coupled ions.

As AFC echoes have been shown to preserve the complete quantum state of single photons for quantum teleportation [101], as well as preserving time-bin qubits for long periods of time by extending to spin-wave storage [102] for other rare earth ions, this is a substantial step towards on-chip cavity-enhanced quantum storage

[1]Using $\kappa_{\text{tot}} = \kappa_c + \kappa$ and $Q = \omega_c/\kappa$, it can be shown for $x = (\kappa_c/\kappa)$, that $Q_l = (1-x)Q_i$.

using erbium.

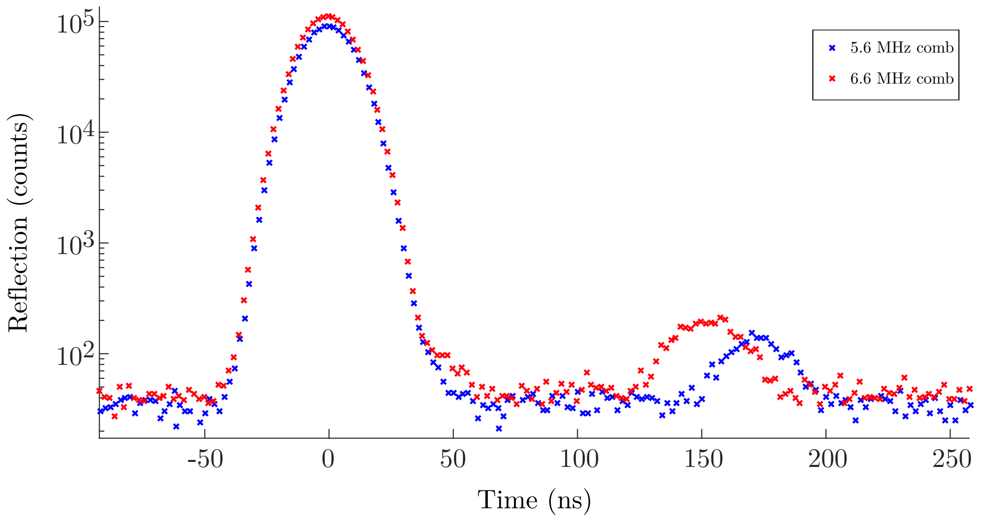


Figure 5.4: Measured atomic frequency comb echoes from two different combs with teeth spacings of 5.6 and 6.6 MHz. The associated storage times should thus be 180 and 150 ns, respectively, which approximately match the observed delays.

Accommodating Requirements of High-Q Cavities

To leverage the substantial improvement in hyperfine lifetimes resulting from high magnetic fields, there are requirements for the cavity that must be considered. Optical manipulation of the hyperfine spin state requires that the cavity can couple to both sets of cross transitions that increase or decrease the nuclear spin by 1. At a magnetic field of 7 T, these transitions are approximately a GHz on either side of the spin-preserving transition. A 2 Ghz splitting between the Δm_I = -1 and +1 transitions would match the full width at half maximum of a cavity with Q = 98,000. As a result, working with higher quality-factor cavities at such high fields might require creative solutions to enable coupling to all three sets of transitions (Δm_I = -1, 0, and +1). Fortunately, it is expected that weaker magnetic fields will also provide long hyperfine lifetimes and coherence times for lower magnetic fields when the sample is at sub-Kelvin temperatures.

5.4 Summary and Future Directions

Coupling rare earth ions in rare earth host crystals to nanoscale optical resonators has been pioneered by the work of the Faraon group [68, 82]. Though prior works have demonstrated coupling to rare earth ions in other host materials, the use of magnetically quiet crystalline hosts is necessary to achieve the long coherence times required for applications of quantum light-matter interfaces. Other work demonstrated mesoscopic optical cavities fabricated using macroscopic machining tools. However, we have conceived, optimized, and demonstrated two architectures for optical cavities, each with mode volumes smaller than 3 cubic microns.

For one architecture, we have demonstrated fabrication of triangular nanobeam cavities with ultra-low mode-volume and highly efficient coupling to free space modes. For the other, we have demonstrated a simple and highly scalable fabrication method for producing cavities and waveguides on chip for coupling to rare earth ensembles. We have demonstrated and thoroughly characterized the coupling between each type of optical cavity and a resonant ensemble of erbium ions. Though an optical quantum memory was not demonstrated, storage of light was achieved using a photon echo in the nanobeam devices, and spectral tailoring of the cavity-coupled ensemble was demonstrated.

Additionally, bulk measurements performed in ^{167}Er:YSO and recent results in the literature show substantial promise and provide a logical extension of the work presented in this thesis. Fabrication of a triangular nanobeam cavity in ^{167}Er:YSO will enable efficient hole burning in the ensemble at high magnetic fields. Use of an additional hyperfine level as a shelving state will allow the creation of an atomic frequency comb with spin-wave storage on a separate hyperfine level to allow storage of infrared photons for hundreds of milliseconds, while the efficiency of storage and read-out will be enhanced by the optical cavity.

Fabrication development could allow the implementation of extensions to the existing devices. Interchanging grating couplers on the hybrid resonators with efficient end-fire couplers would allow the resonators to be tested in the helium-3 fridge at sub-Kelvin temperatures using the existing optics. Furthermore, integrating superconducting nanowire single-photon detectors on chip would substantially improve detection efficiency and increase scalability. Additionally, incorporating electrodes for memory protocols based on the Stark effect provide an additional avenue for implementing on-demand optical quantum memories.

Beyond integration of superconducting detectors and electrodes, fabrication of high quality factor and low mode-volume superconducting microwave cavities is the first step toward using erbium-based efficient on-chip optical to microwave conversion. While optical to microwave conversion is an active field, aiming to bridge the gap between the developed fields of infrared optical communication and superconducting microwave qubits, scalable and efficient conversion of quantum bits with high fidelity has yet to be demonstrated. Often an intermediary excitation, like a phonon is required [103]; however, the hyperfine levels of erbium can circumvent this requirement, as the ion possesses resonances at both the relevant optical and microwave frequencies. To achieve efficient conversion, it is necessary that the ions coupled to the optical cavity are the same ions coupled to the microwave cavity, which likely also requires spatially defined implantation and annealing of erbium.

Detection and manipulation of a single cavity-coupled ion is ongoing within the Faraon group and is more likely to be accomplished in Nd:YVO, where the optical dipole moment is larger and laser stabilization hardware is more developed within our lab.

Also, ongoing work is being done fabricating amorphous silicon hybrid ring resonators on silicon carbide to couple to ensembles of chromium defects or divacancy color centers.

Given that the subfield of rare earth nanophotonics is very young, it is even quite possible that the most important future applications have not yet been proposed or conceived. I have no doubt that interest in coupling rare earth ions to nanophotonic systems will continue to grow, due to the remarkable spectral properties of rare earth ions and the clear benefits of scalable nanofabrication.

BIBLIOGRAPHY

[1] Carl Sagan. *Cosmos*. Cognitive systems monographs. Ballantine Books, 1985. ISBN: 9780345331359.

[2] Paul Benioff. “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines”. In: *Journal of Statistical Physics* 22.5 (May 1980), pp. 563–591. ISSN: 0022-4715. DOI: `10.1007/BF01011339`.

[3] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 0020-7748. DOI: `10.1007/BF02650179`. arXiv: `9508027 [quant-ph]`.

[4] Peter W Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. DOI: `10.1137/S0097539795293172`.

[5] Dan Boneh. “Twenty Years of Attacks on the RSA Cryptosystem”. In: *Notices of the American Mathematical Society* 46.2 (1999), pp. 203–213. ISSN: 00029920. DOI: `10.1.1.525.7995`.

[6] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. New York, NY, USA: Cambridge University Press, 2011. ISBN: 9781107002173.

[7] C. H. Bennett and G. Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*. Vol. 175. 1984, p. 8.

[8] Artur K. Ekert. “Quantum cryptography based on Bell’s theorem”. In: *Physical Review Letters* 67.6 (Aug. 1991), pp. 661–663. ISSN: 0031-9007. DOI: `10.1103/PhysRevLett.67.661`. arXiv: `0911.4171v2`.

[9] Bernard Yurke and David Stoler. “Bells-inequality experiments using independent-particle sources”. In: *Physical Review A* 46.5 (1992), pp. 2229–2234. ISSN: 10502947. DOI: `10.1103/PhysRevA.46.2229`.

[10] H.-J. Briegel et al. “Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication”. In: *Physical Review Letters* 81.26 (1998), pp. 5932–5935. ISSN: 0031-9007. DOI: `10.1103/PhysRevLett.81.5932`. arXiv: `9803056v1 [quant-ph]`.

[11] L.-M. Duan et al. “Long-distance quantum communication with atomic ensembles and linear optics”. In: *Nature* 414.6862 (Nov. 2001), pp. 413–418. ISSN: 0028-0836. DOI: `10.1038/35106500`. arXiv: `0105105 [quant-ph]`.

[12] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Advances in quantum metrology”. In: *Nature Photonics* 5.4 (Apr. 2011), pp. 222–229. ISSN: 1749-4885. DOI: 10.1038/nphoton.2011.35. arXiv: arXiv:1102.2318v1.

[13] Alexander I. Lvovsky, Barry C. Sanders, and Wolfgang Tittel. “Optical quantum memory”. In: *Nature Photonics* 3.12 (2009), pp. 706–714. ISSN: 1749-4885. DOI: 10.1038/nphoton.2009.231. arXiv: 1002.4659.

[14] A. Imamoglu. “High Efficiency Photon Counting Using Stored Light”. In: *Physical Review Letters* 89.16 (2002), p. 163602. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.89.163602. arXiv: 0205196 [quant-ph].

[15] Félix Bussières et al. “Prospective applications of optical quantum memories”. In: *Journal of Modern Optics* 60.18 (June 2013), pp. 1519–1537. ISSN: 0950-0340. DOI: 10.1080/09500340.2013.856482. arXiv: 1306.6904.

[16] International Union of Pure Chemistry and Applied. *Nomenclature of Inorganic Chemistry: IUPAC Recommendations 2005*. 2005, p. 366. ISBN: 0854044388. DOI: 10.1515/ci.2005.27.6.25.

[17] A. J. Freeman and R. E. Watson. “Theoretical investigation of some magnetic and spectroscopic properties of rare-earth ions”. In: *Physical Review* 127.6 (1962), pp. 2058–2075. ISSN: 0031899X. DOI: 10.1103/PhysRev.127.2058.

[18] G. H. Dieke, H. M. Crosswhite, and B. Dunn. “Emission Spectra of the Doubly and Triply Ionized Rare Earths”. In: *Journal of the Optical Society of America* 51.8 (Aug. 1961), p. 820. ISSN: 0030-3941. DOI: 10.1364/JOSA.51.000820.

[19] E. Cicely Ridley. “Self-consistent fields without exchange for Pr^{3+} and Tm^{3+}”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 243.01 (Jan. 1960), p. 422. ISSN: 14698064. DOI: 10.1017/S0305004100034277.

[20] G. H. Dieke and H. M. Crosswhite. “The Spectra of the Doubly and Triply Ionized Rare Earths”. In: *Applied Optics* 2.7 (July 1963), p. 675. ISSN: 0003-6935. DOI: 10.1364/AO.2.000675.

[21] B. R. Judd. “Optical absorption intensities of rare-earth ions”. In: *Physical Review* 127.3 (1962), pp. 750–761. ISSN: 0031899X. DOI: 10.1103/PhysRev.127.750.

[22] G. S. Ofelt. “Intensities of Crystal Spectra of Rare-Earth Ions”. In: *The Journal of Chemical Physics* 37.3 (1962), p. 511. ISSN: 00219606. DOI: 10.1063/1.1701366. arXiv: arXiv:1011.1669v3.

[23] D. L. McAuslan, J. J. Longdell, and M. J. Sellars. “Strong-coupling cavity QED using rare-earth-metal-ion dopants in monolithic resonators: What you can do with a weak oscillator”. In: *Physical Review A* 80.6 (Dec. 2009), pp. 1–9. ISSN: 1050-2947. DOI: `10.1103/PhysRevA.80.062307`. arXiv: `0908.1994v2`.

[24] Manjin Zhong et al. “Optically addressable nuclear spins in a solid with a six-hour coherence time”. In: *Nature* 517.7533 (Jan. 2015), pp. 177–180. ISSN: 0028-0836. DOI: `10.1038/nature14025`. arXiv: `1411.6758`.

[25] R.J. Mears et al. “Low-noise erbium-doped fibre amplifier operating at 1.54μm”. In: *Electronics Letters* 23.19 (1987), p. 1026. ISSN: 00135194. DOI: `10.1049/el:19870719`.

[26] H A Kramers. “Théorie générale de la rotation paramagnétique dans les cristaux”. In: *Proc. Amst. Acad.* 33 (1930), pp. 959–972.

[27] Anatole Abragam and Brebis Bleaney. *Electron Paramagnetic Resonance of Transition Ions*. Oxford University Press, 1970. ISBN: 978-0199651528.

[28] E. Wigner. “Uber die Operation der Zeitumkehr in der Quantenmechanik”. In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen* 31 (1932), pp. 546–559.

[29] Bryan W. Roberts. “Kramers degeneracy without eigenvectors”. In: *Physical Review A* 86.3 (Sept. 2012), p. 034103. ISSN: 1050-2947. DOI: `10.1103/PhysRevA.86.034103`. arXiv: `1208.3889`.

[30] Yongchen Sun et al. “Magnetic g tensors for the i 15/2 4 and i 13/2 4 states of Er3+: Y2 Si O5”. In: *Physical Review B - Condensed Matter and Materials Physics* 77.8 (Feb. 2008), p. 085124. ISSN: 10980121. DOI: `10.1103/PhysRevB.77.085124`.

[31] Valerii Ter-Mikirtychev. *Fundamentals of Fiber Lasers and Fiber Amplifiers*. Vol. 181. Springer Series in Optical Sciences. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-02337-3. DOI: `10.1007/978-3-319-02338-0`.

[32] Thomas Böttger et al. “Spectroscopy and dynamics of Er^{3+}:Y_2SiO_5 at 1.5 μm”. In: *Physical Review B* 74.7 (Aug. 2006), p. 075107. ISSN: 1098-0121. DOI: `10.1103/PhysRevB.74.075107`.

[33] Thomas Böttger et al. “Optical decoherence and spectral diffusion at 1.5 μm in Er^{3+}:Y_2SiO_5 versus magnetic field, temperature, and Er3+ concentration”. In: *Physical Review B - Condensed Matter and Materials Physics* 73.7 (2006), p. 075101. ISSN: 10980121. DOI: `10.1103/PhysRevB.73.075101`.

[34] E. Saglamyurek et al. “An integrated processor for photonic quantum states using a broadband light–matter interface”. In: *New Journal of Physics* 16.6 (June 2014), p. 065019. ISSN: 1367-2630. DOI: `10.1088/1367-2630/16/6/065019`. arXiv: `1402.0481`.

[35] S. Welinski et al. “Effects of disorder on optical and electron spin linewidths in Er3+,Sc3+:Y2SiO5”. In: *Optical Materials* (Sept. 2016). ISSN: 09253467. DOI: 10.1016/j.optmat.2016.09.039.

[36] Gladys H. Fuller. “Nuclear Spins and Moments”. In: *Journal of Physical and Chemical Reference Data* 5.4 (Oct. 1976), pp. 835–1092. ISSN: 0047-2689. DOI: 10.1063/1.555544.

[37] Keith Holliday et al. “Spectral hole burning and holography in an Y_2SiO_5:Pr^{3+} crystal”. In: *Physical Review B* 47.22 (1993), pp. 14741–14752. ISSN: 0163-1829. DOI: 10.1103/PhysRevB.47.14741.

[38] Flurin Könz et al. “Temperature and concentration dependence of optical dephasing, spectral-hole lifetime, and anisotropic absorption in $Eu^{3+}Y_2SiO_5$”. In: *Physical Review B* 68.8 (Aug. 2003), p. 085109. ISSN: 0163-1829. DOI: 10.1103/PhysRevB.68.085109.

[39] Matthias U. Staudt et al. “Investigations of optical coherence properties in an erbium-doped silicate fiber for quantum state storage”. In: *Optics Communications* 266.2 (Oct. 2006), pp. 720–726. ISSN: 00304018. DOI: 10.1016/j.optcom.2006.05.007. arXiv: 0603192 [quant-ph].

[40] Erhan Saglamyurek et al. “Quantum storage of entangled telecom-wavelength photons in an erbium-doped optical fibre”. In: *Nature Photonics* 9.2 (Sept. 2015), pp. 83–87. ISSN: 1749-4885. DOI: 10.1038/nphoton.2014.311. arXiv: 1409.0831.

[41] Miloš Rančić et al. “Coherence time of over a second in a telecom-compatible quantum memory storage material”. In: (Nov. 2016). arXiv: 1611.04315.

[42] L.A. Harris and C.B. Finch. “Mineralogical notes 1493”. In: *The American Minerologist* 50.9 (1965), pp. 1493–1495.

[43] B. A. Maksimov et al. “Crystal Structure of the Y-Oxysilicate Y[SiO4]O”. In: *Kristallografiya* 15.6 (1970), pp. 926–933.

[44] Y. C. Sun. *Spectroscopic Properties of Rare Earths in Optical Materials*. Ed. by Guokui Liu and Bernard Jacquier. Vol. 83. Springer Series in Materials Science. Berlin/Heidelberg: Springer-Verlag, 2005, pp. 379–429. ISBN: 3-540-23886-7. DOI: 10.1007/3-540-28209-2.

[45] C. Li, Ch. Wyon, and R. Moncorge. “Spectroscopic properties and fluorescence dynamics of Er/sup 3+/ and Yb/sup 3+/ in Y/sub 2/SiO/sub 5/”. In: *IEEE Journal of Quantum Electronics* 28.4 (Apr. 1992), pp. 1209–1221. ISSN: 00189197. DOI: 10.1109/3.135248.

[46] Ray Beach et al. “Optical absorption and stimulated emission of neodymium in yttrium orthosilicate”. In: *IEEE Journal of Quantum Electronics* 26.8 (1990), pp. 1405–1412. ISSN: 00189197. DOI: 10.1109/3.59689. arXiv: arXiv:1011.1669v3.

[47] Thomas Böttger et al. “Effects of magnetic field orientation on optical decoherence in Er^{3+}:Y_2SiO_5”. In: *Physical Review B* 79.11 (Mar. 2009), p. 115104. ISSN: 1098-0121. DOI: 10.1103/PhysRevB.79.115104.

[48] C. W. Thiel, W. R. Babbitt, and R. L. Cone. “Optical decoherence studies of yttrium oxyorthosilicate Y_2SiO_5 codoped with Er^{3+} and Eu^{3+} for optical signal processing and quantum information applications at 1.5 microns”. In: *Physical Review B* 85.17 (May 2012), p. 174302. ISSN: 1098-0121. DOI: 10.1103/PhysRevB.85.174302.

[49] S. R. Hastings-Simon et al. “Zeeman-level lifetimes in Er^{3+}:Y_2SiO_5 ”. In: *Physical Review B* 78.8 (Aug. 2008), p. 085410. ISSN: 1098-0121. DOI: 10.1103/PhysRevB.78.085410.

[50] B. Lauritzen et al. “State preparation by optical pumping in erbium-doped solids using stimulated emission and spin mixing”. In: *Physical Review A* 78.4 (Oct. 2008), p. 043402. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.78.043402. arXiv: arXiv:0808.3537v1.

[51] Björn Lauritzen et al. “Telecommunication-Wavelength Solid-State Memory at the Single Photon Level”. In: *Physical Review Letters* 104.8 (Feb. 2010), p. 080502. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.104.080502.

[52] E. Baldit et al. “Identification of Λ-like systems in Er^{3+}:Y_2SiO_5 and observation of electromagnetically induced transparency ”. In: *Physical Review B* 81.14 (Apr. 2010), p. 144303. ISSN: 1098-0121. DOI: 10.1103/PhysRevB.81.144303.

[53] I. D. Abella, N. A. Kurnit, and S. R. Hartmann. “Photon Echoes”. In: *Physical Review* 141.1 (Jan. 1966), pp. 391–406. ISSN: 0031-899X. DOI: 10.1103/PhysRev.141.391.

[54] E.L. Hahn. “Spin Echoes”. In: *Physical Review* 80.4 (1950), pp. 580–594. ISSN: 0031-899X. DOI: 10.1103/PhysRev.80.580.

[55] L. Allen and J.H. Eberly. *Optical Resonance and Two Level Atoms*. Second. New York, NY, USA: Dover Publications, Inc., 1987.

[56] E. L. Hahn. “Free nuclear induction”. In: *Physics Today* 6.11 (Nov. 1953), pp. 4–9. ISSN: 0031-9228. DOI: 10.1063/1.3061075.

[57] Khabat Heshami et al. “Quantum memories: emerging applications and recent advances”. In: *Journal of Modern Optics* 63.20 (Nov. 2016), pp. 2005–2028. ISSN: 0950-0340. DOI: 10.1080/09500340.2016.1148212. arXiv: 1511.04018.

[58] Hugues de Riedmatten et al. “A solid-state light–matter interface at the single-photon level”. In: *Nature* 456.7223 (Dec. 2008), pp. 773–777. ISSN: 0028-0836. DOI: 10.1038/nature07607. arXiv: 0810.0630.

[59] Mikael Afzelius et al. “Multimode quantum memory based on atomic frequency combs”. In: *Physical Review A* 79.5 (May 2009), p. 052329. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.79.052329. arXiv: 0805.4164.

[60] Mikael Afzelius et al. “Demonstration of atomic frequency comb memory for light with spin-wave storage”. In: *Physical Review Letters* 104.4 (2010), pp. 1–4. ISSN: 00319007. DOI: 10.1103/PhysRevLett.104.040503. arXiv: 0908.2309.

[61] Erhan Saglamyurek et al. “Broadband waveguide quantum memory for entangled photons”. In: *Nature* 469.7331 (Jan. 2011), pp. 512–515. ISSN: 0028-0836. DOI: 10.1038/nature09719. arXiv: 1009.0490.

[62] Giacomo Corrielli et al. “Integrated Optical Memory Based on Laser-Written Waveguides”. In: *Physical Review Applied* 5 (May 2016), p. 054013. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.5.054013. arXiv: 1512.09288.

[63] Sara Marzban et al. “Observation of Photon Echoes from Evanescently Coupled Rare-Earth Ions in a Planar Waveguide”. In: *Physical Review Letters* 115.1 (2015), pp. 1–5. ISSN: 10797114. DOI: 10.1103/PhysRevLett.115.013601. arXiv: 1503.0252.

[64] S. Probst et al. “Three-dimensional cavity quantum electrodynamics with a rare-earth spin ensemble”. In: *Physical Review B - Condensed Matter and Materials Physics* 90.10 (2014), pp. 1–5. ISSN: 1550235X. DOI: 10.1103/PhysRevB.90.100404. arXiv: arXiv:1406.3535v1.

[65] Philip Trøst Kristensen, Cole Van Vlack, and Stephen Hughes. “Generalized effective mode volume for leaky optical cavities”. In: *Optics Letters* 37.10 (May 2012), p. 1649. ISSN: 0146-9592. DOI: 10.1364/OL.37.001649. arXiv: 1107.4601.

[66] Mikael Afzelius and Christoph Simon. “Impedance-matched cavity quantum memory”. In: *Physical Review A* 82.2 (Aug. 2010), p. 022310. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.82.022310.

[67] E. M. Purcell. “Spontaneous Emission Probabilities at Radio Frequencies”. In: *Proceedings of the American Physical Society*. Vol. 69. 1946, p. 681. ISBN: 9789729914386.

[68] Tian Zhong et al. “Nanophotonic coherent light–matter interfaces based on rare-earth-doped crystals”. In: *Nature Communications* 6 (Sept. 2015), p. 8206. ISSN: 2041-1723. DOI: 10.1038/ncomms9206.

[69] Tian Zhong et al. “High quality factor nanophotonic resonators in bulk rare-earth doped crystals”. In: *Optics Express* 24.1 (Jan. 2016), p. 536. ISSN: 1094-4087. DOI: 10.1364/OE.24.000536. arXiv: 1512.03947.

[70] Evan Miyazono et al. “Coupling of erbium dopants to yttrium orthosilicate photonic crystal cavities for on-chip optical quantum memories”. In: *Applied Physics Letters* 108.1 (Jan. 2016), p. 011111. ISSN: 0003-6951. DOI: 10.1063/1.4939651.

[71] J. S. Foresi et al. “Photonic-bandgap microcavities in opticalwaveguides”. In: *Nature* 390.6656 (Nov. 1997), pp. 143–145. ISSN: 00280836. DOI: 10.1038/36514.

[72] J. P. Zhang et al. “Nanofabrication of 1-D photonic bandgap structures along a photonic wire”. In: *IEEE Photonics Technology Letters* 8.4 (1996), pp. 491–493. ISSN: 10411135. DOI: 10.1109/68.491093.

[73] O. Painter, J. Vučkovič, and A. Scherer. “Defect modes of a two-dimensional photonic crystal in an optically thin dielectric slab”. In: *Journal of the Optical Society of America B* 16.2 (Feb. 1999), p. 275. ISSN: 0740-3224. DOI: 10.1364/JOSAB.16.000275.

[74] Parag B. Deotare et al. “High quality factor photonic crystal nanobeam cavities”. In: *Applied Physics Letters* 94.12 (Mar. 2009), p. 121106. ISSN: 0003-6951. DOI: 10.1063/1.3107263. arXiv: 0901.4158.

[75] Igal Bayn et al. “Triangular nanobeam photonic cavities in single-crystal diamond”. In: *New Journal of Physics* 13.2 (Feb. 2011), p. 025018. ISSN: 1367-2630. DOI: 10.1088/1367-2630/13/2/025018. arXiv: 1101.1367.

[76] Michael J Burek et al. “Free-Standing Mechanical and Photonic Nanostructures in Single-Crystal Diamond.” In: *Nano letters* (Nov. 2012). ISSN: 1530-6992. DOI: 10.1021/nl302541e.

[77] Michael J. Burek et al. “High quality-factor optical nanocavities in bulk single-crystal diamond”. In: *Nature Communications* 5 (Dec. 2014), p. 5718. ISSN: 2041-1723. DOI: 10.1038/ncomms6718. arXiv: 1408.5973.

[78] Ardavan F. Oskooi et al. “Meep: A flexible free-software package for electromagnetic simulations by the FDTD method”. In: *Computer Physics Communications* 181.3 (2010), pp. 687–702. ISSN: 00104655. DOI: 10.1016/j.cpc.2009.11.008.

[79] S. Mosor et al. “Scanning a photonic crystal slab nanocavity by condensation of xenon”. In: *Applied Physics Letters* 87.14 (2005), pp. 1–3. ISSN: 00036951. DOI: 10.1063/1.2076435.

[80] I. Diniz et al. “Strongly coupling a cavity to inhomogeneous ensembles of emitters: Potential for long-lived solid-state quantum memories”. In: *Physical Review A* 84.6 (Dec. 2011), p. 063810. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.84.063810. arXiv: 1101.1842.

[81] Beatriz Cordero et al. “Covalent radii revisited”. In: *Dalton Transactions* 21 (2008), p. 2832. ISSN: 1477-9226. DOI: 10.1039/b801115j.

[82] Evan Miyazono et al. "Coupling erbium dopants in yttrium orthosilicate to silicon photonic resonators and waveguides". In: *Optics Express* 25.3 (Feb. 2017), p. 2863. ISSN: 1094-4087. DOI: 10.1364/OE.25.002863.

[83] Kevin Luke et al. "Broadband mid-infrared frequency comb generation in a Si_3N_4 microresonator". In: *Optics Letters* 40.21 (Nov. 2015), p. 4823. ISSN: 0146-9592. DOI: 10.1364/OL.40.004823.

[84] F. A. Ponce and D. P. Bour. "Nitride-based semiconductors for blue and green light-emitting devices". In: *Nature* 386.6623 (Mar. 1997), pp. 351–359. ISSN: 0028-0836. DOI: 10.1038/386351a0.

[85] Charles Kittel. *Introduction to Solid State Physics*. 6th. New York: John Wiley, 1986.

[86] Luozhou Li et al. "Nanofabrication on unconventional substrates using transferred hard masks". In: *Scientific Reports* 5 (Jan. 2015), p. 7802. ISSN: 2045-2322. DOI: 10.1038/srep07802.

[87] Cheng-wei Cheng et al. "Epitaxial lift-off process for gallium arsenide substrate reuse and flexible electronics". In: *Nature Communications* 4 (Mar. 2013), p. 1577. ISSN: 2041-1723. DOI: 10.1038/ncomms2583.

[88] A Arbabi et al. "Dielectric metasurfaces for complete control of phase and polarization with subwavelength spatial resolution and high transmission". In: *Nature Nanotechnology* 10.11 (Aug. 2015), pp. 937–943. ISSN: 1748-3387. DOI: 10.1038/nnano.2015.186. arXiv: 1411.1494.

[89] J. Baró et al. "PENELOPE: An algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter". In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 100.1 (May 1995), pp. 31–46. ISSN: 0168583X. DOI: 10.1016/0168-583X(95)00349-5.

[90] V. B. Verma et al. "High-efficiency WSi superconducting nanowire single-photon detectors operating at 2.5 K". In: *Applied Physics Letters* 105.12 (2014), pp. 2013–2016. ISSN: 00036951. DOI: 10.1063/1.4896045. arXiv: 1406.1810.

[91] Franco Gaspari. *Optoelectronics - Materials and Techniques*. Ed. by P. Predeep. InTech, Sept. 2011, pp. 3–26. ISBN: 978-953-307-276-0. DOI: 10.5772/779.

[92] Tian Zhong et al. "High quality factor nanophotonic resonators in bulk rare-earth doped crystals". In: *Optics Express* 24.1 (Jan. 2016), p. 536. ISSN: 1094-4087. DOI: 10.1364/OE.24.000536.

[93] M.A. Rosa et al. "Self-alignment of optical fibers with optical quality end-polished silicon rib waveguides using wet chemical micromachining techniques". In: *IEEE Journal of Selected Topics in Quantum Electronics* 5.5 (1999), pp. 1249–1254. ISSN: 1077260X. DOI: 10.1109/2944.806748.

[94] R. Moosburger et al. “Passive alignment of single-mode fibers to integrated polymer waveguide structures utilizing a single-mask process”. In: *IEEE Photonics Technology Letters* 11.7 (July 1999), pp. 848–850. ISSN: 1041-1135. DOI: 10.1109/68.769728.

[95] Seán M. Meenehan et al. “Silicon optomechanical crystal resonator at millikelvin temperatures”. In: *Physical Review A* 90.1 (July 2014), p. 011803. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.90.011803. arXiv: 1403.3703.

[96] Justin D Cohen, Seán M Meenehan, and Oskar Painter. “Optical coupling to nanoscale optomechanical cavities for near quantum-limited motion transduction”. In: *Optics Express* 21.9 (May 2013), p. 11227. ISSN: 1094-4087. DOI: 10.1364/OE.21.011227. arXiv: 1302.1807.

[97] Björn Lauritzen. “Quantum Memories for Telecommunication Networks”. PhD thesis. 2010, p. 166.

[98] O. Guillot-Noël et al. “Direct observation of rare-earth-host interactions in Er: Y2 Si O5”. In: *Physical Review B* 76.18 (2007), p. 180408. ISSN: 10980121. DOI: 10.1103/PhysRevB.76.180408.

[99] Yu-Hui Chen, Xavier Fernandez-Gonzalvo, and Jevon J. Longdell. “Coupling erbium spins to a three-dimensional superconducting cavity at zero magnetic field”. In: *Physical Review B* 94.7 (Aug. 2016), p. 075117. ISSN: 2469-9950. DOI: 10.1103/PhysRevB.94.075117. arXiv: 1512.03606.

[100] Mikael Afzelius and Christoph Simon. “Impedance-matched cavity quantum memory”. In: *Physical Review A* 82.2 (Aug. 2010), p. 022310. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.82.022310.

[101] Félix Bussières et al. “Quantum teleportation from a telecom-wavelength photon to a solid-state quantum memory”. In: *Nature Photonics* 8.10 (Sept. 2014), pp. 775–778. ISSN: 1749-4885. DOI: 10.1038/nphoton.2014.215. arXiv: 1401.6958.

[102] Mustafa Gündoğan et al. “Solid State Spin-Wave Quantum Memory for Time-Bin Qubits”. In: *Physical Review Letters* 114.23 (June 2015), p. 230501. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.114.230501. arXiv: 1501.03980.

[103] R. W. Andrews et al. “Bidirectional and efficient conversion between microwave and optical light”. In: *Nature Physics* 10.4 (Mar. 2014), pp. 321–326. ISSN: 1745-2473. DOI: 10.1038/nphys2911. arXiv: 1310.5276.

Appendix A

CODE FOR CAVITY MODE SIMULATION AND ANALYSIS

The code in Section A.1 and A.2 were used to simulate the ring and nanobeam cavities, respectively. The code in Section A.3 was used to generate the included figures from the simulation output.[1]

A.1 MEEP Code to Model the Ring

```
; ring-on-YSO-cyl.ctl
; simulates the modes for an amorphous silicon ring on YSO in cylindrical coordinates.
; Evan Miyazono 9/2016

(define-param w 0.4785)       ; width of waveguide
(define-param r 11.0055)      ; outer radius of ring
(define-param h_ring 0.377)      ; ring height

(define-param res 75)         ; resolution

(define-param fcen 0.651)        ; 1536 nm
(define-param df 0.2)
(define-param time 2000)

(set-param! m 118)            ; azimuthal symmetry number

(define-param rpad (ceiling (/ 2 (- fcen df)) ))  ; radial padding between waveguide and ...
    edge of PML
(define-param zpad (ceiling (/ 2 (- fcen df)) ))  ; vertical padding between waveguide and ...
    edge of PML
(define-param dpml 1)  ; thickness of PML

(define n_si 3.48)      ; material making up ring
(define n_YSO 1.8)      ; material above the ring

(define sr (+ r rpad dpml) ) ; radial size (cell is from 0 to sr)
(define sz (+ h_ring (* 2 zpad) (* 2 dpml) ) ) ; vertical size

(set! dimensions CYLINDRICAL)
(set! geometry-lattice (make lattice (size sr no-size sz)))

(set! geometry (list
   (make block                              ; YSO substrate
      (center (/ sr 2) 0 (/ (+ zpad dpml h_ring) 2) ) ; r 0 z (z=0 at center, r=0 at left)
        (size sr infinity (+ zpad dpml) )             ; r phi z
      (material (make dielectric (index n_YSO))))
   (make block                              ; ring
      (center (- r (/ w 2)) 0 0)
      (size w infinity h_ring)
      (material (make dielectric (index n_si))))
) )


```

[1]typesetting of meep (Scheme) and matlab code was accomplished using lstlang0.sty, written by Andreas Stuhlm uller (`https://github.com/stuhlmueller/scheme-listings`) and mcode.sty library, written by Florian Knorn (`www.florian-knorn.com`).

```
(set! pml-layers (list (make pml (thickness dpml))))
(set-param! resolution res) ; simulation resolution

(set! sources (list
               (make source
                 (src (make gaussian-src (frequency fcen) (fwidth df)))
                 (component Ez) (center (+ r (/ w -2.0001)) 0 0))))

(run-sources+
   time
   (after-sources (harminv Ez (vector3 (- r (/ w 2)) 0 (/ h_ring 2)) fcen df))
   (at-beginning output-epsilon)
   (at-end output-efield)
)
```

A.2 MEEP Code to Model the Nanobeam

```
; nanobeam-YSO.ctl
; Tian's cavity scaled to 1536 nm resonance
; initially written by Alex Hartz, modified by Tian Zhong, lastly edited by Evan Miyazono


(define-param n_air 1)      ; index of air holes
(define-param n_YSO 1.8)    ; index for bottom half YSO

(define-param N 20)         ; number of holes on either side of defect
(define cur 0)              ; current hole index

(define-param fcen 0.651)   ; pulse center frequency
(define-param df 0.2)       ; pulse width (in frequency)

(define-param time 1000)    ; running time
(define-param nfreq 50000)  ; number of frequencies at which to compute flux
(define-param w 1)          ; source area

; uncomment the following and run using >mpirun -np 22 meep-mpi nanobeam-YSO.ctl > ...
    resonances_cav.txt &
(define-param dir "resonances_cavity")
(define-param compute-modes? true) ; find resonant modes
(define-param no_holes? false)
(define-param paraFrac .95)

; uncomment the following and run using >mpirun -np 15 meep-mpi nanobeam-YSO.ctl > ...
    transmission_cav.txt &
;(define-param dir "transmission_cavity_hr")
;(define-param compute-modes? false) ; find transmission spectrum
;(define-param no_holes? false)
;(define-param paraFrac .95)

; uncomment the following and run using >mpirun -np 7 meep-mpi nanobeam-YSO.ctl > ...
    transmission_mir.txt &
;(define-param dir "transmission_mirror")
;(define-param compute-modes? false) ; find transmission spectrum
;(define-param no_holes? false)
;(define-param paraFrac 1)

; uncomment the following and run using >mpirun -np 4 meep-mpi nanobeam-YSO.ctl > ...
    transmission_wg.txt &
;(define-param dir "transmission_wg")
;(define-param compute-modes? false) ; find transmission spectrum
;(define-param no_holes? true)
;(define-param paraFrac .95)


```

```
(define-param trenchFrac .432)                      ; trench length/afinal
(define-param trenchDepthFrac .65)                  ; trench depth/nanobeam height at center
(define-param afinal (* 1.7395 .34))                ; final hole spacing 2*.19 ;.35
(define trenchDepth (* (* (tan trenchAngle) (/ midWidth 2) ) trenchDepthFrac) )
(define trenchWidth (* trenchFrac afinal))

(define ainit (* paraFrac afinal))
(define-param spaceInts 7)                          ; number of points in parabolic spacing profile

(define-param midWidth (* 1.7395 0.8))              ; width of top of nanobeam
(define-param trenchWidth (* 1.7395 1.00))
(define-param trenchDegAngle 60)                    ; interior angle of triangular nanobeam
(define trenchAngle (* trenchDegAngle (/ pi 180)))

(define s (/ (/ (+ midWidth trenchWidth) 2) (cos trenchAngle) .5)) ; AEH's super clear ...
    naming scheme
(define deltaSpace (/ (- afinal ainit) (* spaceInts spaceInts) ))

(define-param res 30)

(use-output-directory dir)

(define (findSpace num)
 (cond (
    (<= num spaceInts)
      (set! num (- num 1))
      (+ (* num num deltaSpace) ainit)
    )
    (else
      (* afinal 1)
    )
 )
)

(define (getPos num)
    (cond (
  (= num 1)
    (set! cur (/ ainit 2))
  )
  (else
    (set! cur (+ cur (findSpace num) ))
  )
    )-
    cur
)

(define-param pad 1) ; padding between last hole and PML edge
(define-param dpml 1) ; PML thickness

(define useX (* 2 N afinal))

(define sx (+ useX (* 2 (+ pad dpml)) )) ; size of cell in x direction
(define sy (+ (* 2 trenchWidth) midWidth (* 2 dpml)) )
(define sz (+ s (* 2 dpml) ))

(set! geometry-lattice (make lattice (size sx sy sz)))
(set! default-material airMat)
(define airMat (make dielectric (index n_air)))
(define ysoMat (make dielectric (index n_YSO)))

; Define the substrate, side trenches, and top trenches
(set! geometry
  (append geometry;
    (list
      (make block (center 0 0  (* sz -0.1)) (size sx midWidth (* 0.2 sz))                  ...
          ; YSO substrate
        (material ysoMat))
      (make block (center 0 (/ (+ trenchWidth midWidth) 2) 0)) (size sx trenchWidth s)       ...
          ; side trench 1
        (e1 1 0 0) (e2 0 1 0) (e3 0 (cos trenchAngle) (sin trenchAngle)) (material airMat))
      (make block (center 0 (/ (+ trenchWidth midWidth) -2) 0)) (size sx trenchWidth s)      ...
          ; side trench 2
        (e1 1 0 0) (e2 0 1 0) (e3 0 ( * -1 (cos trenchAngle)) (sin trenchAngle)) (material ...
            airMat))
    )
  )
)

(define tempLoc 0)
(define (addHoles holeN)
    (set! tempLoc (getPos holeN))
    (if (not no_holes?)
      (list
        (set! geometry (append geometry (list
          (make block (center tempLoc 0 0) (size trenchWidth midWidth (* 2 trenchDepth)) ...
              (material airMat))
          (make block (center (* -1 tempLoc) 0 0) (size trenchWidth midWidth (* 2 ...
              trenchDepth)) (material airMat)))))
        (if (< holeN N)
          (addHoles (+ holeN 1) ) )
        )
    )
)
(addHoles 1)

; Run the simulation
(set! pml-layers (list (make pml (thickness dpml))))
(set-param! resolution res)

(if compute-modes?
  (begin
    (set! sources (list
                    (make source
                          (src (make gaussian-src (frequency fcen) (fwidth df)))
                          (component Ey)
                          (center 0 0 (/ (* (tan trenchAngle) (/ midWidth 2)) -3) ) (size 0 ...
                              0 0)
                    )
                  )
    )
    (run-sources+ time
    (at-beginning output-epsilon)
    (after-sources (harminv Ey (vector3 0 0 0) fcen df))
    (at-end output-efield)
    (at-end print (meep-fields-modal-volume-in-box fields (meep-fields-total-volume fields)))
    )
  )
  else
  (begin
    (set! sources (list
                      (make source
                      (src (make gaussian-src (frequency fcen) (fwidth df)))
                      (component Ey)
                      ;(center (+ dpml (* -0.5 sx) (/ pad 2))  ; source at the opposite end
                      (center 0 0 (/ (* (tan trenchAngle) (/ midWidth 2)) -3) ) ; source at ...
                          center
                      (size 0 midWidth))))
      (define trans ; transmitted flux
        (add-flux fcen df nfreq
          (make flux-region
            (center (/ (+ useX pad) 2) 0 (* midWidth - .433))   ; sqrt(3)/4 = .433
            (size 0 midWidth (* midWidth .866))
            (direction X) )))
      (display-fluxes trans) ; print out the flux spectrum
  )
```

```
)
```

A.3 Matlab Code to Analyze the Ring

```
% ring_plotSimulatedPL.m
% imports mode profile from meep (as HDF5) and calculates ring-ensemble coupling
% Written by Evan Miyazono, partly based on code by Andrei Faraon

%% set constants and read meep results
n_Si = 3.41;
n_YSO = 1.8;
lambda_um = 1.536;

eps_fname = 'ring_TM_2D_res_75_m_118_df_0.05_tim_2000-eps-000000000.h5';
e_fname = 'ring_TM_2D_res_75_m_118_df_0.05_tim_2000-e-000060000.h5';

resolution = 75;
interface_index = 404; % YSO is from index 404 to end

fname = ['meep/ring_sims/' eps_fname];
epsilon = hdf5read(fname,'eps');
fname = ['meep/ring_sims/' e_fname];
er = hdf5read(fname,'er.r') + 1i*hdf5read(fname,'er.i');
ep = hdf5read(fname,'ep.r') + 1i*hdf5read(fname,'ep.i');
ez = hdf5read(fname,'ez.r') + 1i*hdf5read(fname,'ez.i');
etot = sqrt(abs(er).^2 + abs(ep).^2 + abs(ez).^2);

dr = 1/resolution;
dz = dr;
r = linspace(0,size(epsilon,1)*dr,size(epsilon,1));
z = linspace(0,size(epsilon,2)*dz,size(epsilon,2)) - size(epsilon,2)*dz/2;

linear_index = find(etot == max(max(etot)));
[max_r, max_z] = ind2sub(size(etot),linear_index);


% meep analysis
figure(1);      % xy plot x, y, and z components of E, and epsilon|E|^2
subplot(2,2,1); imagesc(r,z,epsilon'); title('epsilon')
axis tight;
subplot(2,2,2); imagesc(r,z,abs(ep')); title('ep')
axis tight;
subplot(2,2,3); imagesc(r,z,real(ez')); title('ez')
axis tight;
subplot(2,2,4); imagesc(r,z,epsilon' .* etot'.^2); title('\epsilon * etot^2')
axis tight;

zlims = 300:475;
rlims = 700:900;

fig = figure();
subplot(3,1,1)
yyaxis left
plot(z(zlims),abs(ez(max_r,zlims))/max(abs(ez(max_r,zlims))), 'Color')
ylabel('field')
yyaxis right
plot(z(zlims),abs(epsilon(max_r,zlims)));
set(gca,'XTick',[])
ylabel('\epsilon'); axis tight;

ax1=subplot(3,1,2);
imagesc(z(zlims),r(rlims),real(ez(rlims,zlims)));
set(gca,'XTick',[])
ylabel('r (\mum)'); axis tight;
```

```
colorbar;

ax3=subplot(3,1,3);
imagesc(z(zlims),r(rlims),real(epsilon(rlims,zlims)));
xlabel('z (\mum)' ) % (vacuum, Si, YSO)')
ylabel('r (\mum)'); axis tight;

Et_max = max(max(etot));
Er_max = max(max(abs(er)))/Et_max;
Ep_max = max(max(abs(ep)))/Et_max;
Ez_max = max(max(abs(ez)))/Et_max;

[~,R] = meshgrid(z,r);
dV = dz*2*pi*dr*R;
dU = dV.*epsilon.*etot.^2;
U_tot = sum(sum(dU));
U_YSO = sum(sum(dU(:,interface_index:end)));

% Standing wave mode volume and YSO mode volume
if Er_max < (Ep_max^2 + Ez_max^2)
    % factor of 0.5 for azimuthal sinusoidal variation
    V = 0.5 * U_tot/(max(max(epsilon.*(abs(ez).^2 + abs(ep).^2))));
    V_YSO = U_YSO/(max(max(epsilon.*(abs(ez).^2 + abs(ep).^2))));
    disp('looks like a TM mode')
else
    V = U_tot/(max(max(epsilon.*(abs(er).^2))));
    V_YSO = U_YSO/(max(max(epsilon.*(abs(er).^2))));
    disp('looks like a TE mode')
end
disp( ['volume = ' num2str(V/(lambda_um^3/(n_YSO * n_Si^2))) ' cubic wavelengths (in aSi)'] )
disp( ['volume = ' num2str(V) ' um^3'] )
disp( ['volume in YSO = ' num2str(V_YSO) ' um^3'] )
disp( ['fraction of field energy in YSO = ' num2str(U_YSO/U_tot)] )

% ion density in ions/um^3 for 0.02% Er:YSO (factor of 1/2 for site1/site2)
ion_density = 1/2 * 0.0002 * (2*4.44/286 / 1e4^3 * 6.022e23);
disp(['total number of ions in the YSO mode volume: ' num2str(V_YSO * ion_density)])

disp( ['Max E field in YSO/max E field in aSi ratio = ' ...
    num2str(max(max(abs(ez(:,interface_index:end)))) / max(max(abs(ez))))] )

% truncate simulation at interface_index for integrals over the ions
Ez_YSO_abs_ratio = abs(ez(:,interface_index:end)) / max(max(abs(ez)));
dV_YSO = dV(:,interface_index:end);

%% plot Eratio histogram
values = Ez_YSO_abs_ratio(:).^2;
weights = ion_density * dV_YSO(:);% .* Ez_YSO_abs_ratio(:).^2;
edges = linspace(min(values), max(values), 75);
[~, bin] = histc(values,edges);
count = accumarray(bin,weights);
figure;
bar(edges, count)
set(gca,'YScale','log')
xlim([-0.01 1.01]);
title('ring')

%% matlab computation constants
c=3*10^8;
hbar=1.0546*10^-34;
vac_permit = 8.854e-12;
lambda=lambda_um*10^-6;

% YSO parameters
omega_1536 = 2*pi*3e8/lambda; % in hertz
mu = 2.07e-32;          % in Cm from McAuslan "...do with a weak oscillator"

% resonator parameters
```

```
Q=80400;    % from fit (FitCavityAndIonsScript_gtotal)
V_mode = V*1e-18;           % in cubic meters
kappa=2*pi*c/(lambda*Q);    % energy decay in 2pi Hz
T_1 = 0.0114;               % in seconds
detuning = 2*pi*0e9;        % in 2pi Hz

% cavity effective index calculated by weighing epsilon by the |E|^2
g0 = mu / n_Si * sqrt(omega_1536/(2*hbar*vac_permit * V_mode));                 % in 2pi Hz

g_map = g0 * Ez_YSO_abs_ratio;
g_tot = sqrt(sum(sum(ion_density*dV_YSO.*(g_map/(2*pi)).^2)));
disp(['g_total is ' num2str( g_tot/1e6 ) ' MHz' ])
g_eff = 2*pi*sqrt(sum(sum(Ez_YSO_abs_ratio.^2.*(g_map/(2*pi)).^2)) / sum(sum( ...
    Ez_YSO_abs_ratio.^2 )) );
disp(['g_0 is ' num2str( g0/1e6 ) ' (2pi)MHz' ])
disp(['g_effective is ' num2str( g_eff/1e6 ) ' (2pi)MHz' ])
disp(['effective number of ions (g_tot/g_eff): ' num2str(g_tot / g_eff)])

chi_L = ((n_YSO^2 + 2)/3)^2;
T_spon = 3*vac_permit*hbar*lambda^3/(8*pi^2*n_YSO*mu^2*chi_L);
Fp_max = 3/(4*pi^2) * lambda^3/(n_Si^2*n_YSO*chi_L) * Q/V_mode * T_1/T_spon;

%% iterate and fit all PL files
directory = 'plfromaug4th/'; % Have to be in folder plfromaug4th
filelist=dir([directory 'tune*PL*.phu']);
% detuned frequency measured via wavemeter after PL
meas_detunings = [[195116.78,195117.75,195118.80]-195125.11,...
                  [195116.74,195115.74,195117.75]-195123.30,...
                  [195116.81,195115.80,195117.77]-195122.43,...
                  [195116.75,195115.78,195117.76]-195121.39,...
                  [195116.78,195115.77,195117.76]-195120.67,...
                  [195116.75,195117.75,195115.76]-195119.60,...
                  [195116.75,195115.76,195117.78]-195118.61,...
                  [195116.88,195117.77,195115.74]-195115.94,...
                  [195116.90,195117.73,195115.75]-195114.62,...
                  [195116.80,195115.77,195117.75]-195113.73]*1e9;

dt = 0.004096; %ms
startPL = 27;
stopPL = startPL-1+7200/3;%
binsize = 25;

PL_time = (0:binsize:(stopPL-startPL))*dt*1e-3;   % time axis in seconds
single_exp_lifetimes = zeros(size(filelist)); % to save the single exp fit

meas_decay = zeros(length(meas_detunings), length(PL_time)); % the PL data

% selected specific files for higher SNR (subset of file_list)
high_snr_filenums = [2,5:6,8:9,11:27,29:30];
% use file 27 (detuning 1.13 GHz) for fig. 4a
for fileindex = high_snr_filenums
    %extract counts, crop and bin PL curve
    filename=filelist(fileindex).name;
    [counts, times] = get_counts_from_PHU([directory filename],0);
    counts = counts(:,1)';
    data_x = times(startPL:binsize:stopPL)-times(startPL);  % in seconds
    data_y = counts(startPL:stopPL);
    data_y = sum(reshape(data_y,[binsize,length(data_x)]));

    decay_rate_map = 1/T_1 + g_map.^2*kappa / ((kappa/2)^2 + meas_detunings(fileindex)^2);
    PL = zeros('like',PL_time);
    Ez_ratio_threshold = 0.001; % skip weakly coupled ions (doesn't change result)
    for voxel_index = 1:numel(Ez_YSO_abs_ratio)
        if Ez_YSO_abs_ratio(voxel_index) > Ez_ratio_threshold
            Fp_voxel = Fp_max * Ez_YSO_abs_ratio(voxel_index)^2;
            PL = PL + dV_YSO(voxel_index) .* Fp_voxel/(Fp_voxel+1) * ...
                exp(-PL_time*decay_rate_map(voxel_index)) * ...
                decay_rate_map(voxel_index);
        end
    end
    PL = PL/max(PL);
    meas_decay(fileindex, :) = PL;

    % plot fits
    figure(fileindex); hold on;
    plot(data_x*1e3,data_y,'x')

    % fit PL_sim
    f_simdecay=@(params,PL_time) (params(1)*PL + params(2));
    [sim_params,~,~,CovB_sim,~,~]=nlinfit(data_x,data_y,f_simdecay,  ...
        [240,400],statset('MaxIter', 1e6));
    plot(PL_time*1e3, f_simdecay(sim_params, PL_time));

    % fit single exp
    f_expdecay=@(params,PL_time) (params(1)*exp(-PL_time/params(3)) + params(2));
    [exp_params,~,~,CovB_exp,~,~]=nlinfit(data_x,data_y,f_expdecay, ...
        [240,400,0.011],statset('MaxIter', 1e6));
    plot(PL_time*1e3, f_expdecay(exp_params, PL_time));

    xlabel('time (ms)')
    ylabel('PL (arb)')
    set(gca,'yscale','log');
    legend({'data','simulated decay','single exp'});
end

%% generate simulated PL intensity vs detuning contour plot
binsize = 10;
sim_detunings = linspace(-50,50,200)*1e9;
PL_time = (0:binsize:(stopPL-startPL))*dt*1e-3;   % in seconds
sim_decay = zeros(length(sim_detunings),length(PL_time));

for detuning_index = 1:length(sim_detunings)
    decay_rate_map = 1/T_1 + 2*g_map.^2*kappa / (kappa^2 + sim_detunings(detuning_index)^2);
    PL = zeros('like',PL_time);
    Ez_ratio_threshold = 0.001;
    for voxel_index = 1:numel(Ez_YSO_abs_ratio)
        if Ez_YSO_abs_ratio(voxel_index) > Ez_ratio_threshold
            Fp_voxel = Fp_max * Ez_YSO_abs_ratio(voxel_index)^2;
            PL = PL + dV_YSO(voxel_index) .* Fp_voxel/(Fp_voxel+1) * ...
                exp(-PL_time*decay_rate_map(voxel_index)) * ...
                decay_rate_map(voxel_index); % to normallize integral(PL) to 1
        end
    end
    PL = PL/max(PL);
    sim_decay(detuning_index, :) = PL;
end

inversion_proportion = cumsum(sim_decay,2)./meshgrid(sum(sim_decay,2),PL_time)';
figure();
imagesc(PL_time*1e3, sim_detunings/1e9, 1-inversion_proportion);
xlabel('time (ms)')
ylabel('detuning (GHz)')
```

Appendix B

CODE FOR RUNNING OPTICAL CHARACTERIZATION EXPERIMENTS

B.1 Tektronix 5014 Arbitrary Waveform Generator

```
% Tektronix_AWG5041.m
%
% Interface object to control a Tektronix AWG5014B arbitrary waveform generator
% Built from a few sources
%   http://www1.tek.com/forum/viewtopic.php?f=6&t=3217
%   Tektronix AWG5014 programmer manual from online saved in fgen_libraries
%
% The AWG can run in continuous mode or sequence mode.
% Continuous mode involves using a single waveform per channel.
% Sequence mode
% Functions that work with only one mode should
%
% Make sure to turn off the AWG5014 VXI-11 server (and probably GPIB) and
% enable LAN communication using port 4000 on the AWG5014 itself (its
% system menu in the program that runs the outputs)
%
% If a session gets interrupted and the AWG refuses to connect to matlab,
% turn LAN communication off and then on again (on the AWG5014 itself) and
% then try reconnecting.
%
% If the marker vector doesn't load, check out and debug the
% create_waveform command - I had to do some sketchy shit to make it work.
%
% usage:
%     Awg_instance = Tektronix_AWG5014('169.254.178.97', 1064)
%     Awg_instance.clear();
%     Awg_instance.create_waveform('hole_amp', hole_amp_waveform, mark_start, mark_end);
%     Awg_instance.set_channel_waveform(1,'hole_amp');
%     Awg_instance.set_sampling_rate(sample_rate*1000);
%     Awg_instance.set_repetition_rate(1/(total_time*1e-3));
%     Awg_instance.start_output([1,2]);
%
% ETM 20151105


classdef (ConstructOnLoad = true) Tektronix_AWG5014 < handle
    properties (SetAccess = private)
        awg_tcpip;
        buffer_size;

        channel_has_waveform;
        channel_output_on;
        sampling_rate_limits;
    end
    methods
        function obj = Tektronix_AWG5014(address, out_buffer_size)

            % check naively that address is IP-like (doesn't check <255)
            if regexp(address,'^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$')
                instr_address = address;
            else
                instr_address  = '169.254.178.97';
                warning('provided address invalid, using 169.254.178.97');
            end

            if out_buffer_size < 0
                obj.buffer_size = 1064;
            else
                obj.buffer_size = out_buffer_size;
            end

            obj.awg_tcpip = tcpip(instr_address, 4000,'OutputBufferSize', obj.buffer_size);

            fopen(obj.awg_tcpip);     % Connect to instrument object, obj.

            % obj.clear();
            obj.channel_has_waveform = [0,0,0,0];
            obj.channel_output_on = [0,0,0,0];
            obj.sampling_rate_limits = [1e7, 10e9];
        end
        %% control
        % reset waveforms and settings
        function clear(obj)
            flushinput(obj.awg_tcpip);
            flushoutput(obj.awg_tcpip);
            fprintf(obj.awg_tcpip,'*rst;');
            fprintf(obj.awg_tcpip,'*cls;');
        end
      function close(obj)
            warning('off','TekAWG5014:channelcheck')
            stop_output(obj,[1,2,3,4]);
            warning('on','TekAWG5014:channelcheck')
            fclose(obj.awg_tcpip);    % Disconnect all objects.
        end
        % turn output 'on' and run for channels named in a vector of length
        % 'channels' lists channel numbers to turn on; ex. [1,2] or [4,2,3]
        function start_output(obj,channels)
            for channel = 1:4
                % only turn on the passed in channels with waveforms loaded
                if ~isempty(find(channels==channel,1))
                    if obj.channel_has_waveform(channel)
                        fprintf(obj.awg_tcpip,['OUTP' num2str(channel) ' ON']);
                        obj.channel_output_on(channel) = 1;
                    else
                        warning('TekAWG5014:channelcheck',['channel ' num2str(channel) ...
                                 ' does not have a waveform loaded... dummy']);
                    end
                end
            end
            fprintf(obj.awg_tcpip, ':awgcontrol:run;');
        end
        % stop running and turn output 'off' for channels named in a vector of length
        % 'channels' lists channel numbers to turn off; ex. [1,2] or [4,2,3]
        function stop_output(obj,channels)
            fprintf(obj.awg_tcpip, ':awgcontrol:stop;');
            for channel = 1:4
                if ~isempty(find(channels==channel,1))
                    if obj.channel_output_on(channel)
                        fprintf(obj.awg_tcpip,['OUTP' num2str(channel) ' OFF']);
                        obj.channel_output_on(channel) = 0;
                    else
                        warning('TekAWG5014:channelcheck',['channel ' num2str(channel) ...
                                 ' was not on.  What kind of shit are you trying to pull?']);
                    end
                end
            end
        end
        % prevents execution of new commands until pending commands are executed
        function finish_current_command(obj)
            fprintf(obj.awg_tcpip, '*wai');
        end
        %% getters
        % lists both user defined and predefined waveforms
        function waveform_name_list = get_waveform_names(obj)
            fprintf(obj.awg_tcpip, 'wlist:size?');
            num_names = str2double(fscanf(obj.awg_tcpip));
            waveform_name_list = cell(num_names,1);
            for name_index = 1:num_names
                fprintf(obj.awg_tcpip, ['wlist:name? ' num2str(name_index-1)]);
                waveform_name_list{name_index} = fscanf(obj.awg_tcpip);
            end
        end
        % gets waveform name - not on sequence mode
        function waveform_name = get_channel_waveform_name(obj, channel_num)
            fprintf(obj.awg_tcpip, [':source' num2str(channel_num) ':waveform?']);
            waveform_name = fscanf(obj.awg_tcpip);
        end
        function voltage_amplitude = get_channel_amplitude(obj, channel)
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':voltage?']);
            voltage_amplitude = fscanf(obj.awg_tcpip);
        end
        function sampling_rate_limits = get_sampling_rate_limits(obj)
            sampling_rate_limits = obj.sampling_rate_limits;
        end
        function num_steps = get_sequence_num_steps(obj)
            fprintf(obj.awg_tcpip,':SEQuence:length?');
            num_steps = str2double(fscanf(obj.awg_tcpip));
        end
        function total_points = get_total_loaded_points(obj)
            total_points = 0;
            names = get_waveform_names(obj);
            for waveform_index = 26:length(names) % first 25 are preloaded
                total_points = total_points + ...
                    get_waveform_length(obj,names{waveform_index});
            end
        end
        function length = get_waveform_length(obj, name)
            fprintf(obj.awg_tcpip,['WLISt:WAVeform:LENgth? ' name]);
            length = str2double(fscanf(obj.awg_tcpip));
        end
        function message = get_error_message(obj)
            fprintf(obj.awg_tcpip,':SYSTem:ERRor?');
            message = str2double(fscanf(obj.awg_tcpip));
        end

        %% setters
        % Send a waveform and markers to store in memory under the given
        % name.  Note: waveform_vector is scaled so that the largest value
        % is either 1 or -1 (and a warning is thrown) while marker_vector
        % must be either 0 or 1
        function create_waveform(obj, waveform_name, waveform_vector, marker_vector1, ...
            marker_vector2)
            if isempty(waveform_vector)
                error([waveform_name ' is empty']);
            end
            if max(waveform_vector)>1 || min(waveform_vector)<-1
                waveform_vector = waveform_vector / max(abs(waveform_vector));
                disp(['range is [' num2str(max(waveform_vector)) ',' ...
                    num2str(min(waveform_vector)) ']'])
                warning('rescaling waveform_vector to be between -1 and 1')
            end

            if length(waveform_vector) ~= length(marker_vector1) || ...
                    length(waveform_vector) ~= length(marker_vector2)
                warning(['waveform and marker vectors must have equal ' ...
                         'length...  the markers are all set low until '...
                         'you get your shit together'],0);
                marker_vector1 = zeros(size(waveform_vector));
                marker_vector2 = marker_vector1;
            end

            waveform_length = length(waveform_vector);
            single_vector = single(waveform_vector);
            % reshape so each point is a byte column
            binary_waveform = reshape(typecast(single_vector,'uint8'),[4,waveform_length]);

            % encode marker 1 bits to bit 6
            m1 = bitshift(uint8(logical(marker_vector1)),6); %check dec2bin(m1(2),8)
            % encode marker 2 bits to bit 7
            m2 = bitshift(uint8(logical(marker_vector2)),7); %check dec2bin(m2(2),8)
            % merge markers
            marker_vector = m1 + m2; %check dec2bin(marker_vector(2),8)

            % add on the marker data
            binary_waveform_wmarker = vertcat(binary_waveform,marker_vector);

            % reshape to 4 wave bytes then 1 marker byte repeating
            binary_waveform_wmarker = reshape(binary_waveform_wmarker,1,5*waveform_length);

            bytes = num2str(length(binary_waveform_wmarker));
            header = ['#' num2str(length(bytes)) bytes];

            fprintf(obj.awg_tcpip,[':wlist:waveform:new "' ...
                waveform_name '",' num2str(waveform_length) ',REAL;']);
            send_long_command(obj,[':wlist:waveform:data "' ...
                waveform_name '",' header binary_waveform_wmarker ';']);

            % now set the marker data alone, because sometimes it fucks up
            % the end of the waveform when marker 2 is high at the end
            marker_bytes = num2str(length(marker_vector));
            marker_header = ['#' num2str(length(marker_bytes)) marker_bytes];
            send_long_command(obj,[':wlist:waveform:marker:data "' ...
                waveform_name '",' marker_header marker_vector+2^5 ';']);
            % OH MY FUCKING GOD, THIS HAS TO BE THE WORST BUG I'VE EVER
            % ENCOUNTERED. THE ABOVE LINES SHOULD RESET THE MARKER TO WHAT
            % IT ALREADY IS BUT FOR SOME REASON THIS FIXES THE WAVEFORM?!?!
            % and don't ask me what the FUCK that 2^5 does exactly
            % but it fixes it somehow.

            % check names afterward to confirm upload
            % nice idea, but it takes way too long to run
%             if find(strcmp(get_waveform_names(obj),waveform_name))
%                 error(['well, shit... uploading ' waveform_name ' failed'])
%             end
        end
        function set_channel_waveform(obj, channel, waveform_name)
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':waveform "' waveform_name '";']);
            obj.channel_has_waveform(channel) = 1;
        end
        function set_channel_voltage_range(obj, channel, V_range)
            amplitude = range(V_range);
            offset = mean(V_range);
            if range(V_range) > 4.5
                warning(['voltage range on channel ' num2str(channel) ' is too high'])
            end
            if mean(V_range) < -2.25
                warning(['voltage offset on channel ' num2str(channel) ' is too low'])
            end
            if mean(V_range) > 2.25
                warning(['voltage offset on channel ' num2str(channel) ' is too high'])
            end
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':voltage ' num2str(amplitude)]);
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':voltage:offset ' ...
                num2str(offset) ]);
        end
        function set_channel_amp_volts(obj, channel, amplitude)
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':voltage ' num2str(amplitude)]);
        end
        function set_channel_offset_volts(obj, channel, offset)
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':voltage:offset ' ...
                num2str(offset) ]);
        end
        function set_marker_out_range(obj, channel, marker_num, marker_high, marker_low)
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':marker' num2str(marker_num) ...
                ':voltage:high ' num2str(marker_high) ]);
            fprintf(obj.awg_tcpip,[':source' num2str(channel) ':marker' num2str(marker_num) ...
                ':voltage:low ' num2str(marker_low) ]);
        end
        function set_repetition_rate(obj, rep_rate)
            fprintf(obj.awg_tcpip,[':awgcontrol:rrate ' num2str(rep_rate)]);
        end
        function set_sampling_rate(obj, samp_rate)
            if samp_rate < obj.sampling_rate_limits(1)
                warning(['sample rate too low.  setting to ' ...
                    num2str(obj.sampling_rate_limits(1))])
                samp_rate = num2str(obj.sampling_rate_limits(1));
            else if samp_rate > obj.sampling_rate_limits(2)
                warning(['sample rate too high.  setting to ' ...
                    num2str(obj.sampling_rate_limits(2))])
                samp_rate = num2str(obj.sampling_rate_limits(2));
                end
            end
            fprintf(obj.awg_tcpip,['source:frequency ' num2str(samp_rate) ]);
        end
        function set_sequence_mode_on(obj, bool)
            if bool
                fprintf(obj.awg_tcpip,'AWGControl:RMODe sequence');
            else
                fprintf(obj.awg_tcpip,'AWGControl:RMODe continuous');
            end
        end
        % assign a waveform to a channel sequence step
        function set_channel_waveform_seq_step(obj, channel, step, waveform_name)
            fprintf(obj.awg_tcpip,['SEQuence:ELEMent' num2str(step) ':WAVeform' ...
                num2str(channel) ' "' waveform_name '";']);
            obj.channel_has_waveform(channel) = 1;
        end
        function set_channel_seq_step_loop_num(obj, step, loop_num)
            fprintf(obj.awg_tcpip,[':SEQuence:ELEMent' num2str(step) ':loop:count ' ...
                num2str(loop_num) ';']);
        end
        function set_sequence_num_steps(obj,numsteps)
            fprintf(obj.awg_tcpip,[':SEQuence:length ' num2str(numsteps) ]);
        end
        function set_sequence_step_goto(obj,gofrom_step,goto_step)
            fprintf(obj.awg_tcpip,[':SEQuence:ELEMent' num2str(gofrom_step) ':GOTO:STATe 1']);
            fprintf(obj.awg_tcpip,[':SEQuence:ELEMent' num2str(gofrom_step) ':GOTO:INDex ' ...
                num2str(goto_step) ]);
        end

        function delete_waveform_by_name(obj, name)
            % note, the name 'all' will delete all user waveforms
            fprintf(obj.awg_tcpip,[':wlist:waveform:delete' name]);
        end
        function clear_all_sequence_steps(obj)
            obj.set_sequence_num_steps(0);
            obj.channel_has_waveform = [0,0,0,0];
        end

        % to be used when sending commands over 1064 characters long
        function send_long_command(obj,command_string)
            bytes = length(command_string);
            if obj.buffer_size >= bytes
               % might have to make this fwrite for proper formatting?
               fprintf(obj.awg_tcpip,command_string);
            else
                % write buffer_size blocks till what's left is <buffer_size
                for i = 1:obj.buffer_size:bytes-obj.buffer_size
                    fwrite(obj.awg_tcpip,command_string(i:i+obj.buffer_size -1));
                end
                fwrite(obj.awg_tcpip,command_string(i+obj.buffer_size:end));
                obj.finish_current_command();
            end
        end
    end
end
```

B.2 Sequence Loader

```
% Sequence_loader.m
%
% Object that interfaces with the Tektronix_AWG5014 object to more cleanly
% upload and keep track of various pulse steps in a sequence to eliminate
% unnecessary loading of long steps (breaks down and loops flat steps).
%
% Constructor takes an active Tektronix_AWG5014.m object and each run_***
% takes a fully populated <sequence> struct.  A <sequence> struct's
% parameters vary by function, and are outlined in each function.
%
% Functions starting with "run_" are experiment-level commands, which take
% a sequence object containing the relevant sequence parameters.  These use
% helper functions with names formatted as "make_***_step".  The helper
% functions create and upload analog and digital waveforms for the relevant
% channels as steps.  If the step doesn't exist, the make___step commands
% will create it, upload it, and add it to the list of loaded waveforms; it
% does NOT add it to the current sequence.  Once waveforms are uploaded,
% they must be added to either the channel (in continuous mode) or the step
% (in sequence mode). Within the AWG, waveforms are stored individually with
% 1 analog + 2 digital. Because waveforms must be the same number of points,
% they are stored together in Sequence_loader in "step" objects, all created
% by functions that call the function make_generic_step.
%
%   step objects possess:
%       name - (theoretically) unique identifier string
%       amp - analog output for channel amp_channel_num
%       freq - analog output for channel freq_channel_num
%       output - analog output for channel output_channel_num
%       scan_marker - digital output
%       sync_marker - digital output
%       MEMS_marker - digital output
%
% The function plot_current(downsample_factor) will plot the loaded steps.
% A downsample factor of ~100 is recommended, as the number of total points
% will likely be large.
%
%   list of experiment-level "run" functions
%       run_hole_burn
%       run_trench_burn_scan
%       run_trench_burn
%       run_stepNburn_afc_scan
%       run_stepNburn_afc_echo
%       run_accumulated_afc
%       run_accumulated_afc_echo
%       run_echo
%   list of step creation functions
%       frac_n_make_burn_step
%       frac_n_make_burn_trench_step
%       make_full_burn_step
%       make_full_burn_step_trench
%       make_burn_ramp_step
%       make_burn_trench_ramp_step
%       make_burn_pair_step_loop
%       make_stepNburn_step_loop
%       make_readout_step
%       make_sync_trigger_step
%       make_scan_n_triggersync_step
%       frac_n_make_wait_step
%       make_flat_step
%       make_cosine_step
%       make_generic_step
%   list of utility functions
%       is_name_loaded
%       get_step_from_name
%       upload_step
%       append_to_sequence
%       close_sequence_loop
%       plot_current
%       concat_waveform
%       check_rounding
%
% output chanels are as follows:
%    amplitude waveform - voltage to the AOM
%       - marker 1: input_marker - high when input (AOM) is >0 (for RF switch)
%       - marker 2: sync_marker - triggers timing card (timeharp) for scan or read
%    freq waveform - voltage to the laser piezo
%       - marker 1: scan_marker - triggers on the middle of the scan range
%       - marker 2: MEMS_marker - a pulse at beginng and end of read or scan pulse
%    output waveform - voltage to output AOM
%       - marker 1: output_marker - high when output is 1
%       - marker 2: 1-input_marker - high when input is 0
%
% usage:
%     Awg_instance = Tektronix_AWG5014('169.254.22.43', 1064);
%     sequence_loader = Sequence_loader(Awg_instance);
%
%     afc_sequence.burn_amplitude = 0.04;
%     afc_sequence.burn_time = 1;
%     afc_sequence.burn_freq_rise_time = 1;
%     afc_sequence.teeth_range = [-.2,.2];
%     afc_sequence.num_teeth = 5;
%     afc_sequence.wait_times = [0.5 100 0.5 30];
%     afc_sequence.num_burn_loops = 50;
%     afc_sequence.input_rise_time = 0;
%     afc_sequence.out_rise_time = 0;
%     afc_sequence.freq_rise_time = 5;
%     afc_sequence.wait_freq_offset = 0;
%     afc_sequence.hole_freq_offset = 0;
%     afc_sequence.read_amplitude = 1;
%     afc_sequence.read_time = 0.00004;
%     afc_sequence.num_read_loops = 200;
%     afc_sequence.MEMS_rise_time = 0;
%     afc_sequence.MEMS_trigger_time = 0.1;
%     afc_sequence.block_readout_step = 0;
%
%     sequence_loader.run_stepNburn_afc_echo(afc_sequence)
%
% ETM,IC 20160203

classdef (ConstructOnLoad = true) Sequence_loader < handle
    properties (SetAccess = private)
        Awg_instance;

        % loaded_steps contains all loaded 'step' structs, uniquely named,
        % which includes the step type (i.e. burn, wait) and relevant params.
        loaded_steps;
        % approximate number of loaded points, to determine when to clear
        % saved waveforms
        total_loaded_points;
        % channel numbers for input, output, and frequency waveforms
        freq_channel_num;
        input_channel_num;
        output_channel_num;
        % constant to determine the output value for output "on"
        output_channel_on;

        sample_rate = 50000; %200000; % 1e6; %  samples per s
        trigger_time = 1/10;  % in ms (MEMS switch intermittently misses shorter)
        max_flat_time;
        verbose;
        current_steps;
        current_sequence;

        MEMS_block_output = 0;
        MEMS_pass_output = 1;
        output_block = -1;
        output_pass = 1;

    end
    methods
        function obj = Sequence_loader(Awg_instance)
            Awg_instance.clear();
            obj.Awg_instance = Awg_instance;
            obj.loaded_steps = {};
            obj.total_loaded_points = 0;
            obj.Awg_instance.set_sequence_mode_on(true);

            obj.output_channel_num = 2;
            obj.input_channel_num = 3;
            obj.freq_channel_num = 4;
            obj.max_flat_time = 10; % in ms
            obj.verbose = true;

            % set the MEMS marker amplitude
            % set_marker_out_range(obj, channel, marker_num, marker_high, marker_low)
            obj.Awg_instance.set_marker_out_range(obj.freq_channel_num, 2, 2, 0)
        end

        function sequence = get_current_sequence(obj)
            sequence = obj.current_sequence;
        end

        function clean_for_new_sequence(obj)
            if obj.total_loaded_points > 129.6e6 - 50e6  % AWG only holds 129.6e6 total points
                if obj.verbose
                    disp(['*** Total number of stored waveform points ('...
                          num2str(obj.total_loaded_points) ...
                          ') is too large; clearing waveforms ***'])
                end
                obj.Awg_instance.delete_waveform_by_name('all')

                obj.loaded_steps = {};
                obj.total_loaded_points = 0;
            end

            obj.Awg_instance.clear_all_sequence_steps();
            obj.Awg_instance.set_sequence_mode_on(true);
            obj.Awg_instance.set_sampling_rate(obj.sample_rate*1000);
        end

%% full experiment-level "run" commands

        % burn a spectral hole and scan it after a wait period
        function run_hole_burn(obj, sequence)
            % A _sequence_ struct is presumed to contain:
            %     total_time - total time of the run (next 3 values + end buffer)
            %     burn_time - time duration of hole burn
            %     wait_time - time duration of wait between burn end and midscan
            %     scan_time - time for linear region of scan
            %     input_rise_time - rise time for AOM signal (reduce ringing)
            %     freq_rise_time - rise time for laser piezo (reduce ringing)
            %     burn_amplitude - amplitude (to AOM) of burn pulse between 0,1
            %     scan_amplitude - amplitude (to AOM) of scan pulse between 0,1
            %     hole_freq_offset - piezo offset during hole burn
            %     wait_freq_offset - piezo offset during wait time
            %     scan_freq_range - 1x2 vector with min and max of piezo scan range

            obj.clean_for_new_sequence();

            obj.output_channel_on = 1; % turn on output channel loadings for this

            intermission_time = sequence.total_time - sequence.burn_time...
                                - sequence.wait_time - sequence.scan_time;

            if sequence.input_rise_time > sequence.wait_time || ...
                    sequence.freq_rise_time > sequence.wait_time
                error('increase wait time or decrease rise time')
            end
            if 2*sequence.input_rise_time > intermission_time || ...
                    2*sequence.freq_rise_time > intermission_time
                error('increase total time or decrease rise time')
            end

            % burn
            burn_steps = obj.frac_n_make_burn_step(sequence.freq_rise_time,...
                            sequence.input_rise_time,sequence.burn_time,...
                            sequence.burn_amplitude,sequence.hole_freq_offset,...
                            sequence.wait_freq_offset,obj.output_block,obj.MEMS_block_output);
            obj.append_to_sequence(burn_steps);

            % wait
            wait_steps = obj.frac_n_make_wait_step(sequence.wait_time - ...
                    2*sequence.freq_rise_time, sequence.wait_freq_offset, obj.output_pass, ...
                        obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps);

            % scan
            scan_step = obj.make_scan_n_triggersync_step(sequence.freq_rise_time,...
                            sequence.input_rise_time, sequence.scan_time,...
                            sequence.scan_freq_range, sequence.scan_amplitude,...
                            sequence.wait_freq_offset);
            obj.append_to_sequence(scan_step);

            % wait2
            wait_steps2 = obj.frac_n_make_wait_step(intermission_time - ...
                    2*sequence.freq_rise_time, sequence.wait_freq_offset, obj.output_pass, ...
                        obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps2);


            % run stuff
            obj.current_steps = [burn_steps, wait_steps, scan_step, wait_steps2];
            obj.current_sequence = sequence;
            obj.Awg_instance.finish_current_command();
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num);
            end
        end

        % burn a wide spectral hole (trench) and scan it after a wait period
        % A _sequence_ struct is presumed to contain:
        %     total_time - total time of the run (next 3 values + end buffer)
        %     burn_time - time duration of hole burn
        %     wait_time - time duration of wait between burn end and midscan
        %     scan_time - time for linear region of scan
        %     input_rise_time - rise time for AOM signal (reduce ringing)
        %     freq_rise_time - rise time for laser piezo (reduce ringing)
        %     burn_amplitude - amplitude (to AOM) of burn pulse between 0,1
        %     scan_amplitude - amplitude (to AOM) of scan pulse between 0,1
        %     hole_freq_offset - piezo offset during hole burn
        %     wait_freq_offset - piezo offset during wait time
        %     scan_freq_range - 1x2 vector with min and max of piezo scan range
        function run_trench_burn_scan(obj, sequence)

            obj.clean_for_new_sequence();

            obj.output_channel_on = 1; % turn on output channel loadings for this

            intermission_time = sequence.total_time - sequence.burn_time...
                                - sequence.wait_time - sequence.scan_time;

            if sequence.wait_time < 4*obj.trigger_time
                error('Wait time before scan pulse is too short for MEMS switch trigger pulse')
            end
            if sequence.input_rise_time > sequence.wait_time
                error('increase wait time or decrease rise time')
            end
            if sequence.input_rise_time > intermission_time
                error('increase total time or decrease rise time')
            end

            % burn
            burn_steps = obj.frac_n_make_burn_trench_step(sequence.freq_modulation_period, ...
                                sequence.input_rise_time,sequence.burn_time,...
                                sequence.burn_amplitude,sequence.trench_fraction);
            obj.append_to_sequence(burn_steps);

            % wait (keep MEMS off for some additional time to block burn pulse)
            wait_steps_MEMS = obj.frac_n_make_wait_step(4*obj.trigger_time, 0, ...
                obj.output_block, obj.MEMS_block_output);
            obj.append_to_sequence(wait_steps_MEMS);

            wait_steps = obj.frac_n_make_wait_step(sequence.wait_time - 4*obj.trigger_time - ...
                    sequence.freq_rise_time, 0, obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps);

            % scan
            scan_step = obj.make_scan_n_triggersync_step(sequence.freq_rise_time, ...
                sequence.input_rise_time, ...
                            sequence.scan_time, sequence.scan_freq_range, ...
                                sequence.scan_amplitude,0);
            obj.append_to_sequence(scan_step);

            % wait2
            wait_steps2 = obj.frac_n_make_wait_step(intermission_time - 4*obj.trigger_time - ...
                    sequence.freq_rise_time,0,obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps2);
            obj.append_to_sequence(wait_steps_MEMS);

            % run stuff
            obj.current_steps = [burn_steps, wait_steps_MEMS, wait_steps, scan_step, ...
                wait_steps2, wait_steps_MEMS];
            obj.current_sequence = sequence;
            obj.Awg_instance.finish_current_command();
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num);
            end
        end

        % burn a wide spectral hole (trench) and read it out using heterodyne interference
        % with a pulse it after a wait period
        % A _sequence_ struct is presumed to contain:
        %     total_time - total time of the run (next 3 values + end buffer)
        %     burn_time - time duration of hole burn
        %     wait_time - time duration of wait between burn end and midscan
        %     scan_time - time for linear region of scan
        %     input_rise_time - rise time for AOM signal (reduce ringing)
        %     freq_rise_time - rise time for laser piezo (reduce ringing)
        %     burn_amplitude - amplitude (to AOM) of burn pulse between 0,1
        %     scan_amplitude - amplitude (to AOM) of scan pulse between 0,1
        %     hole_freq_offset - piezo offset during hole burn
        %     wait_freq_offset - piezo offset during wait time
        %     scan_freq_range - 1x2 vector with min and max of piezo scan range
        function run_trench_burn(obj, sequence)

            obj.clean_for_new_sequence();

            obj.output_channel_on = 1; % turn on output channel loadings for this

            intermission_time = sequence.total_time - sequence.burn_time...
                                - sequence.wait_time - sequence.read_time;

            if sequence.wait_time < 4*obj.trigger_time
                error('Wait time before scan pulse is too short for MEMS switch trigger pulse')
            end

            if sequence.input_rise_time > sequence.wait_time
                error('increase wait time or decrease rise time')
            end
            if sequence.input_rise_time > intermission_time
                error('increase total time or decrease rise time')
            end

            % burn
            if sequence.burn_time <= 0
                burn_steps = [];
            else
                burn_steps = ...
                    obj.frac_n_make_burn_trench_step(sequence.freq_modulation_period, ...
                                sequence.input_rise_time,sequence.burn_time,...
                                sequence.burn_amplitude,sequence.trench_fraction);
                obj.append_to_sequence(burn_steps);
            end
            % wait (keep MEMS off for some additional time to block burn pulse)
            wait_steps_MEMS = obj.frac_n_make_wait_step(4*obj.trigger_time, 0, ...
                obj.output_block, obj.MEMS_block_output);
            obj.append_to_sequence(wait_steps_MEMS);

            wait_steps = obj.frac_n_make_wait_step(sequence.wait_time - 4*obj.trigger_time, ...
                0, obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps);
            % readout
            if sequence.block_readout_step == 0
                readout_output = obj.output_pass;
            else
                readout_output = obj.output_block;
            end

            readout_step = make_readout_step(obj, sequence.read_amplitude,...
                            sequence.read_time, 0, 0,readout_output, 1);
            obj.append_to_sequence(readout_step);

            % wait2 (turn MEMS off for some additional time to block burn pulse)
            wait_steps2 = obj.frac_n_make_wait_step(intermission_time -4*obj.trigger_time, ...
                0, obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(wait_steps2);

            obj.append_to_sequence(wait_steps_MEMS);
            % run stuff
            obj.current_steps = [burn_steps, wait_steps_MEMS, wait_steps, readout_step, ...
                wait_steps2, wait_steps_MEMS];
            obj.current_sequence = sequence;
            obj.Awg_instance.finish_current_command();
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num);
            end
        end

        % burn and then scan an afc at 0 frequency detuning of the laser
        % by burning each tooth individually (using make_stepNburn_step_loop)
        % A _sequence_ struct is presumed to contain:
        %     burn_amplitude - heights of pulses
        %     burn_times - width of pulses in ms
        %     wait_time - wait time between pulses in ms
        %     teeth_range - [min,max] of linear region of scan
        %     num_teeth - number of teeth to be burned in the comb
        %     num_burn_loops - number of times to sweep over the full comb
        %     input_rise_time - rise time for input signal (reduce ringing)
        %     out_rise_time - rise time for output signal (reduce ringing)
        %     freq_rise_time - rise time for frequency signal (reduce ringing)
        %     wait_freq_offset - piezo offset during wait time
        %     hole_freq_offset - piezo offset during burn time
        %     scan_amplitude - amplitude (to AOM) of scan pulse between 0,1
        %     scan_freq_range - 1x2 vector with min and max of piezo scan range
        function run_stepNburn_afc_scan(obj, sequence)

            obj.clean_for_new_sequence();
            obj.output_channel_on = 0;

            if sequence.wait_times(2) < obj.trigger_time
                error('Wait time before scan pulse is too short for MEMS switch trigger pulse')
            end
            if sequence.wait_times(3) < obj.trigger_time
                error('Wait time after scan pulse is too short for MEMS switch trigger pulse')
            end

            % if 2*sequence.input_rise_time > sequence.wait_times(1)
            %     error('increase wait between burn pulses or decrease rise time')
            % end
            % if 2*sequence.input_rise_time > sequence.wait_times(3) || ...
            %         2*sequence.freq_rise_time > sequence.wait_times(3)
            %     error('increase prescan wait time or decrease rise time')
            % end
            % if 2*sequence.input_rise_time > sequence.wait_times(4) || ...
            %         2*sequence.freq_rise_time > sequence.wait_times(4)
            %     error('increase postscan wait time or decrease rise time')
            % end

            % burn
            stepNburn_step = obj.make_stepNburn_step_loop(sequence.burn_amplitude,...
                                sequence.burn_time, sequence.wait_times(1), sequence.num_teeth,...
                                sequence.num_burn_loops, sequence.burn_freq_rise_time,...
                                sequence.input_rise_time, sequence.hole_freq_offset,...
                                sequence.teeth_range, sequence.wait_freq_offset);
            obj.append_to_sequence(stepNburn_step);

            % wait
            prescan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(2), ...
                                        sequence.wait_freq_offset, obj.output_pass, ...
                                            obj.MEMS_pass_output);
            obj.append_to_sequence(prescan_wait_steps);

            % scan
            scan_step = obj.make_scan_n_triggersync_step(sequence.freq_rise_time,...
                            sequence.input_rise_time, sequence.scan_time,...
                            sequence.scan_freq_range, sequence.scan_amplitude,...
                            sequence.wait_freq_offset);
            obj.append_to_sequence(scan_step);

            % wait
            postscan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(3), ...
                                        sequence.wait_freq_offset, obj.output_pass, ...
                                            obj.MEMS_pass_output);
            obj.append_to_sequence(postscan_wait_steps);


            % run stuff
            obj.current_steps = [stepNburn_step, prescan_wait_steps, ...
                                    scan_step, postscan_wait_steps];
            obj.current_sequence = sequence;
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num)
            end
        end

        % burn an afc and then probe an afc echo at 0 frequency detuning of the laser
        % by burning each tooth individually (using make_stepNburn_step_loop)
        % A _sequence_ struct is presumed to contain:
        %     burn_amplitude - heights of pulses
        %     burn_times - width of pulses in ms
        %     wait_time - wait time between pulses in ms
        %     teeth_range - [min,max] of linear region of scan
        %     num_teeth - number of teeth to be burned in the comb
        %     num_burn_loops - number of times to sweep over the full comb
        %     input_rise_time - rise time for input signal (reduce ringing)
        %     out_rise_time - rise time for output signal (reduce ringing)
        %     freq_rise_time - rise time for frequency signal (reduce ringing)
        %     wait_freq_offset - piezo offset during wait time
        %     hole_freq_offset - piezo offset during burn time

        %     read_amplitude - amplitude (to AOM) of read pulse between 0,1
        %     read_time - time duration of the read pulse
        %     num_read_loops - number of reads per AFC
        function run_stepNburn_afc_echo(obj, sequence)

            obj.clean_for_new_sequence();
            obj.output_channel_on = 0;

            if sequence.wait_times(2) < obj.trigger_time
                error('Wait time before scan pulse is too short for MEMS switch trigger pulse')
            end
            if sequence.wait_times(4) < obj.trigger_time
                error('Wait time after scan pulse is too short for MEMS switch trigger pulse')
            end

            % burn
            stepNburn_step = obj.make_stepNburn_step_loop(sequence.burn_amplitude,...
                            sequence.burn_time, sequence.wait_times(1), sequence.num_teeth,...
                            sequence.num_burn_loops, sequence.burn_freq_rise_time,...
                            sequence.input_rise_time, sequence.hole_freq_offset,...
                            sequence.teeth_range, sequence.wait_freq_offset);
            obj.append_to_sequence(stepNburn_step);

            % wait

            prescan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(2), ...
                                        sequence.wait_freq_offset, obj.output_pass, ...
                                            obj.MEMS_pass_output);
            obj.append_to_sequence(prescan_wait_steps);

            % read
            if sequence.block_readout_step == 0
                readout_output = obj.output_pass;
            else
                readout_output = obj.output_block;
            end

            readout_step = obj.make_readout_step(sequence.read_amplitude,...
                                sequence.read_time, sequence.wait_times(3), ...
                                sequence.hole_freq_offset, readout_output, sequence.num_read_loops);
            obj.append_to_sequence(readout_step);

            % wait
            postscan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(4), ...
                sequence.wait_freq_offset, obj.output_pass, obj.MEMS_pass_output); % 1 for ...
                output
            obj.append_to_sequence(postscan_wait_steps);


            % run stuff
            obj.current_steps = [stepNburn_step, prescan_wait_steps, ...
                readout_step, postscan_wait_steps];
            obj.current_sequence = sequence;
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num)
            end

            % set MEMS switch output
            obj.Awg_instance.set_marker_out_range(4, 2, 2.5, 0);
        end

        % burn afc using pulse pairs and then scan it at 0 frequency detuning of the laser
        % A _sequence_ struct is presumed to contain:
        %     burn_amplitudes - height of [first,second] pulses
        %     burn_times - width of [first,second] pulses in ms
        %     wait_times - wait time after [pulse1,pulse2,before scan,after scan] in ms
        %     scan_time - time for linear region of scan
        %     num_burn_loops - number of times to send in burn pairs
        %     input_rise_time - rise time for input signal (reduce ringing)
        %     out_rise_time - rise time for output signal (reduce ringing)
        %     freq_rise_time - rise time for frequency signal (reduce ringing)
        %     wait_freq_offset - piezo offset during wait time
        %     hole_freq_offset - piezo offset during burn time
        %     scan_amplitude - amplitude (to AOM) of scan pulse between 0,1
        %     scan_freq_range - 1x2 vector with min and max of piezo scan range
        function run_accumulated_afc(obj, sequence)

            obj.clean_for_new_sequence();
            obj.output_channel_on = 0; % turn off output channel loadings for this

            assert(length(sequence.burn_times)==2 && ...
                   length(sequence.burn_amplitudes)==2 && ...
                   length(sequence.wait_times)==4);
            if 2*sequence.input_rise_time > sequence.wait_times(1)
                error('increase wait between burn pulses or decrease rise time')
            end
            if 2*sequence.input_rise_time > sequence.wait_times(3) || ...
                    2*sequence.freq_rise_time > sequence.wait_times(3)
                error('increase prescan wait time or decrease rise time')
            end
            if 2*sequence.input_rise_time > sequence.wait_times(4) || ...
                    2*sequence.freq_rise_time > sequence.wait_times(4)
                error('increase postscan wait time or decrease rise time')
            end

            % burn
            burn_pair_loop_step = obj.make_burn_pair_step_loop(sequence.burn_amplitudes,...
                            sequence.burn_times, sequence.wait_times(1:2), ...
                            sequence.num_burn_loops, sequence.freq_rise_time, ...
                            sequence.input_rise_time, sequence.hole_freq_offset, ...
                            sequence.wait_freq_offset);
            obj.append_to_sequence(burn_pair_loop_step);

            % wait
            prescan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(3), ...
                                                       sequence.wait_freq_offset, ...
                                                           obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(prescan_wait_steps);

            % scan
            scan_step = obj.make_scan_n_triggersync_step(sequence.freq_rise_time,...
                            sequence.input_rise_time, sequence.scan_time,...
                            sequence.scan_freq_range, sequence.scan_amplitude,...
                            sequence.wait_freq_offset);
            obj.append_to_sequence(scan_step);

            % wait
            postscan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(4), ...
                                                       sequence.wait_freq_offset, 0, ...
                                                           obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(postscan_wait_steps);


            % run stuff
            obj.current_steps = [burn_pair_loop_step, prescan_wait_steps, ...
                                 scan_step, postscan_wait_steps];
            obj.current_sequence = sequence;
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num)
            end
        end


        % burn an afc using pulse pairs and then probe an afc echo at 0 frequency detuning ...
            of the laser
        % A _sequence_ struct is presumed to contain:
        %     burn_amplitudes - height of [first,second] pulses
        %     burn_times - width of [first,second] pulses in ms
        %     wait_times - wait time after [pulse1,pulse2,before scan,each readouts,after ...
            scan] in ms
        %     scan_time - time for linear region of scan
        %     num_burn_loops - number of times to send in burn pairs
        %     input_rise_time - rise time for input signal (reduce ringing)
        %     out_rise_time - rise time for output signal (reduce ringing)

        %     read_amplitude - amplitude (to AOM) of read pulse between 0,1
        %     read_time - time duration of the read pulse
        %     num_read_loops - number of reads per AFC
        function run_accumulated_afc_echo(obj, sequence)

            obj.clean_for_new_sequence();
            obj.output_channel_on = 0; % turn off output channel loadings for this

            assert(length(sequence.burn_times)==2 && ...
                   length(sequence.burn_amplitudes)==2 && ...
                   length(sequence.wait_times)==5);
            if sequence.wait_times(3) < obj.trigger_time
                warning([ 'Not enough time to trigger the MEMS switch between burn and'...
                 ' readout pulses. Wait must currently be >' num2str(obj.trigger_time) ' ms'])
            end

            % burn
            burn_pair_loop_step = obj.make_burn_pair_step_loop(sequence.burn_amplitudes,...
                            sequence.burn_times, sequence.wait_times(1:2), ...
                            sequence.num_burn_loops, 0, sequence.input_rise_time, 0, 0);

            obj.append_to_sequence(burn_pair_loop_step);

            % wait
            prescan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(3), ...
                                        0, obj.output_pass, obj.MEMS_pass_output);
            obj.append_to_sequence(prescan_wait_steps);

            % read
            if sequence.block_readout_step == 0
                readout_output = obj.output_pass;
            else
                readout_output = obj.output_block;
            end

            readout_step = obj.make_readout_step(sequence.read_amplitude,...
                            sequence.read_time, sequence.wait_times(4), ...
                            0, readout_output, sequence.num_read_loops);
            obj.append_to_sequence(readout_step);

            % wait
            postscan_wait_steps = obj.frac_n_make_wait_step(sequence.wait_times(5),...
                             0, obj.output_pass, obj.MEMS_pass_output); % 1 for output
            obj.append_to_sequence(postscan_wait_steps);

            % run stuff
            obj.current_steps = [burn_pair_loop_step, prescan_wait_steps, ...
                readout_step, postscan_wait_steps];
            obj.current_sequence = sequence;
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num,obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num)
            end
        end

        % run an echo at 0 frequency detuning of the laser
        % keeps MEMS switch open at all times (output_channel not used)
        %
        % A _sequence_ struct is presumed to contain:
        %     burn_amplitudes - height of [first,second,(third)] pulses
        %         amplitudes(3)=0 or length(amplitudes)=2 sets 2 pulse echo
        %     total_time - total time of the run (sum of next 3 values + end buffer)
        %     burn_times - width of [first,second,(third)] pulses in ms
        %         burn_times(3)=0 or length(burn_times)=2 sets 2 pulse echo
        %     wait12 - time between end of first and start of second pulse
        %     waitT - time between end of second and start of third pulse
        %     input_rise_time - rise time for input signal (reduce ringing)
        %     output_rise_time - rise time for output signal (reduce ringing)
        function run_echo(obj, sequence)
            obj.clean_for_new_sequence();
            obj.output_channel_on = 1;  % turn on output channel loadings for this

            assert( sequence.input_rise_time < sequence.wait12 || ...
                max(sequence.input_rise_time, sequence.wait12) == 0); % otherwise you can't ...
                    turn things on and off
            assert( sequence.output_rise_time < sequence.wait12 || ...
                max(sequence.output_rise_time, sequence.wait12) == 0 ); % otherwise you ...
                    won't see the echo
            assert( length(sequence.burn_amplitudes)==length(sequence.burn_times) );

            obj.check_rounding([sequence.burn_times sequence.wait12 sequence.waitT]);

            intermission_time = sequence.total_time - sum(sequence.burn_times)...
                - sequence.wait12 - sequence.waitT;

            sync_trigger_step = obj.make_sync_trigger_step(0, 1);
            obj.append_to_sequence(sync_trigger_step);

            if sequence.burn_times(1)>0
                first_burn_step = obj.frac_n_make_burn_step(0, sequence.input_rise_time,...
                                        sequence.burn_times(1),sequence.burn_amplitudes(1),0,0, ...
                                            obj.output_pass,obj.MEMS_pass_output);
                obj.append_to_sequence(first_burn_step);
            end

            wait_tau12_step = obj.frac_n_make_wait_step(sequence.wait12, 0, obj.output_pass, ...
                obj.MEMS_pass_output);
            obj.append_to_sequence(wait_tau12_step);

            if sequence.burn_times(2)>0
                second_burn_step = obj.frac_n_make_burn_step(0, sequence.input_rise_time,...
                                        sequence.burn_times(2),sequence.burn_amplitudes(2),0,0, ...
                                            obj.output_pass,obj.MEMS_pass_output);
                obj.append_to_sequence(second_burn_step);
            end

            listen_step = obj.frac_n_make_wait_step(intermission_time, 0, obj.output_pass, ...
                obj.MEMS_pass_output);

            if length(sequence.burn_amplitudes) == 3 ...
                     && length(sequence.burn_times) == 3 && sequence.burn_times(3)~=0
                assert( 2*sequence.input_rise_time < sequence.waitT );
                wait_T_step = obj.frac_n_make_wait_step(sequence.waitT, ...
                    sequence.wait_freq_offset, obj.output_pass, obj.MEMS_pass_output);
                obj.append_to_sequence(wait_T_step);

                third_burn_step = obj.frac_n_make_burn_step(0, sequence.input_rise_time,...
                                        sequence.burn_times(3),sequence.burn_amplitudes(3),0,0, ...
                                            obj.output_pass,obj.MEMS_pass_output);
                obj.append_to_sequence(third_burn_step);
                obj.current_steps = [sync_trigger_step, first_burn_step, wait_tau12_step, ...
                    second_burn_step, wait_T_step, third_burn_step, listen_step];
            else
                if sequence.burn_times(1)==0
                    obj.current_steps = [sync_trigger_step, wait_tau12_step, ...
                        second_burn_step, listen_step];
                elseif sequence.burn_times(2)==0
                    obj.current_steps = [sync_trigger_step, first_burn_step, ...
                        wait_tau12_step, ...
                        listen_step];
                else
                    obj.current_steps = [sync_trigger_step, first_burn_step, ...
                        wait_tau12_step, ...
                        second_burn_step, listen_step];
                end
            end

            obj.append_to_sequence(listen_step);

            % run stuff
            obj.current_sequence = sequence;
            obj.close_sequence_loop();
            obj.Awg_instance.start_output([obj.input_channel_num, obj.freq_channel_num]);
            if obj.output_channel_on
                obj.Awg_instance.start_output(obj.output_channel_num);
            end
        end

%% step creation functions

```

```
        % make a hole burn step in 3 or 4 parts, looping the middle section if
        % necessary (and possible having some remainder burn step), and calling
        % make_burn_ramp_step for the start and end
        function burn_steps = frac_n_make_burn_step(obj, freq_rise_time, ...
                                input_rise_time,burn_time,burn_amplitude,...
                                hole_freq_offset,wait_freq_offset,output,MEMS)
            % conditionally split the wait time into blocks of length obj.max_flat_time
            if burn_time > 2*obj.max_flat_time
                num_loops = floor(burn_time / obj.max_flat_time);
                remainder = mod(burn_time, obj.max_flat_time);

                if freq_rise_time > 0
                    burn_pre = make_burn_ramp_step(obj, freq_rise_time, ...
                                        input_rise_time,burn_amplitude,...
                                        hole_freq_offset,wait_freq_offset,output,MEMS,1);
                    burn_pre.num_loops = 1;

                    burn_post = make_burn_ramp_step(obj, freq_rise_time, ...
                                        input_rise_time,burn_amplitude,...
                                        hole_freq_offset,wait_freq_offset,output,MEMS,0);
                    burn_post.num_loops = 1;
                end
                                                    % this 0 for output channel ->
                burn_loop = make_flat_step(obj, obj.max_flat_time, burn_amplitude, ...
                                        hole_freq_offset,output,MEMS);
                burn_loop.num_loops = num_loops;

                if remainder > 0       % don't make an empty wait step
                                                    % this 0 for output channel ->
                    burn_remainder = make_flat_step(obj, remainder, burn_amplitude,...
                                        hole_freq_offset,output,MEMS);
                    burn_remainder.num_loops = 1;
                    if freq_rise_time > 0
                        burn_steps = [burn_pre burn_loop burn_remainder burn_post];
                    else
                        burn_steps = [burn_loop burn_remainder];
                    end
                else
                    if freq_rise_time > 0
                        burn_steps = [burn_pre burn_loop burn_post];
                    else
                        burn_steps = burn_loop;
                    end
                end
            else
                burn_steps = make_full_burn_step(obj, freq_rise_time, ...
                                    input_rise_time,burn_time,burn_amplitude,...
                                    hole_freq_offset,wait_freq_offset,output,MEMS);
            end
        end

        % make a trench burn step in 3 or 4 parts, looping the middle section if
        % necessary (and possible having some remainder burn step), and calling
        % and calling make_burn_trench_ramp_step for the start and end
        function burn_steps = frac_n_make_burn_trench_step(obj,  freq_mod_period, ...
                                input_rise_time,burn_time,burn_amplitude,...
                                trench_fraction)
            % conditionally split the wait time into blocks of length obj.max_flat_time
            actual_burn_time=round(burn_time/(freq_mod_period/2))*(freq_mod_period/2);

            if actual_burn_time ~= burn_time
               warning(['burn time input was ', num2str(burn_time), ...
                   ' burn time actually used is ' num2str(actual_burn_time)])
            end

            burn_time=actual_burn_time;
            freq_rise_time=freq_mod_period/2;

            if actual_burn_time < freq_mod_period
                error('burn time must be >= frequency modulation period')
            end

            num_loops = floor((burn_time-freq_mod_period)/freq_mod_period);

            odd_half_periods=mod(burn_time/(freq_mod_period/2),2);

            burn_pre = make_burn_trench_ramp_step(obj, freq_rise_time, ...
                                                  input_rise_time,burn_amplitude,...
                                                  trench_fraction,1,0);
            burn_pre.num_loops = 1;

            if burn_time > 3*freq_rise_time
                burn_loop = make_cosine_step(obj,freq_mod_period,...
                    freq_mod_period,trench_fraction,burn_amplitude);
                burn_loop.num_loops = num_loops;
            else
                burn_loop = [];
            end

            if odd_half_periods

                burn_post = make_burn_trench_ramp_step(obj, freq_rise_time, ...
                                                      input_rise_time,burn_amplitude,...
                                                      trench_fraction,0,1);
                burn_extra_halfcos = make_cosine_step(obj,freq_mod_period/2,...
                freq_mod_period,trench_fraction,burn_amplitude);
                burn_extra_halfcos.num_loops = 1;

                burn_steps = [burn_pre burn_loop burn_extra_halfcos burn_post];
            else
                burn_post = make_burn_trench_ramp_step(obj, freq_rise_time, ...
                                                      input_rise_time,burn_amplitude,...
                                                      trench_fraction,0,0);
                burn_post.num_loops = 1;
                burn_steps = [burn_pre burn_loop burn_post];
            end
        end

        % make a hole burn step, with rise and fall in one step
        function burn_steps = make_full_burn_step(obj, freq_rise_time, ...
                                    input_rise_time,burn_time,burn_amplitude,...
                                    hole_freq_offset,wait_freq_offset,output,MEMS)
            name = ['brn' num2str(freq_rise_time) 'fqrs,' ...
                                num2str(input_rise_time) 'inrs,' ...
                                num2str(burn_time) 'tm,' ...
                                num2str(burn_amplitude) 'brnpw,' ...
                                num2str(hole_freq_offset) 'hofs,' ...
                                num2str(wait_freq_offset) 'wofs'];
            freq_rise_samples = freq_rise_time * obj.sample_rate;
            input_rise_samples = input_rise_time * obj.sample_rate;
            burn_samples = round(burn_time * obj.sample_rate);
            amp_wave = [zeros(1,freq_rise_samples-input_rise_samples),...
                    burn_amplitude * (0:1/(input_rise_samples-1):1),...
                    burn_amplitude * ones(1,burn_samples),...
                    burn_amplitude * (1:-1/(input_rise_samples-1):0),...
                    zeros(1,freq_rise_samples-input_rise_samples)];
            freq_wave = [flat2flat_transition(freq_rise_samples, ...
                    wait_freq_offset, hole_freq_offset),...
                    hole_freq_offset*ones(1,burn_samples),...
                    flat2flat_transition(freq_rise_samples, ...
                    hole_freq_offset,wait_freq_offset)];

            % zero for output_marker, scan_marker, sync_marker, MEMS_marker
            Os = zeros(size(amp_wave));
            burn_steps = obj.make_generic_step(name, amp_wave, freq_wave, ...
                                                 output+Os, Os, Os, MEMS+Os);
            burn_steps.num_loops = 1;
        end

        % make a trench burn step in 3 parts, calling
        % make_burn_trench_ramp_step for the start and end
        function burn_steps = make_full_burn_step_trench(obj, freq_mod_period, ...
                                input_rise_time,burn_time,burn_amplitude,...
                                trench_fraction)
            name = ['brn' num2str(freq_mod_period) 'fqmp,' ...
                            num2str(input_rise_time) 'inrs,' ...
                            num2str(burn_time) 'tm,' ...
                            num2str(burn_amplitude) 'brnpw,' ...
                            num2str(trench_fraction) 'trfrc,'];

            actual_burn_time=round(burn_time/(freq_mod_period/2))*(freq_mod_period/2);

            if actual_burn_time ~= burn_time
               warning(['burn time input was ', num2str(burn_time), ...
                   ' burn time actually used is ' num2str(actual_burn_time)])
            end
            burn_time=actual_burn_time;
            odd_half_periods=mod(burn_time/(freq_mod_period/2),2);

            input_rise_samples = input_rise_time * obj.sample_rate;
            burn_samples = (burn_time * obj.sample_rate);
            freq_mod_period_samples=freq_mod_period * obj.sample_rate;
            amp_wave = [burn_amplitude * (0:1/(input_rise_samples-1):1),...
                    burn_amplitude * ones(1,burn_samples),...
                    burn_amplitude * (1:-1/(input_rise_samples-1):0)];

            if odd_half_periods
            freq_wave = [zeros(1,input_rise_samples),...
                    flat2flat_transition(freq_mod_period_samples/2, ...
                    0, trench_fraction),...
                    cos_freq_mod(trench_fraction,freq_mod_period_samples, ...
                        burn_samples-freq_mod_period_samples),...
                    flat2flat_transition(freq_mod_period_samples/2, ...
                    -trench_fraction,0),...
                    zeros(1,freq_rise_samples-input_rise_samples)];
            else
            freq_wave = [zeros(1,input_rise_samples),...
                    flat2flat_transition(freq_mod_period_samples/2, ...
                    0, trench_fraction),...
                    cos_freq_mod(trench_fraction,freq_mod_period_samples, ...
                        burn_samples-freq_mod_period_samples),...
                    flat2flat_transition(freq_mod_period_samples/2, ...
                    trench_fraction,0),...
                    zeros(1,freq_rise_samples-input_rise_samples)];
            end

            % zero for output_marker, scan_marker, sync_marker, MEMS_marker
            Os = zeros(size(amp_wave));
            burn_steps = obj.make_generic_step(name, amp_wave, freq_wave, ...
                                            obj.output_block+Os, Os, Os, ...
                                                obj.MEMS_block_output+Os);
            burn_steps.num_loops = 1;
        end

        % the rise or falling side of a pulse used to burn a spectral hole. Frequency
        % is constant while amplitude increases or decreases according to up_not_down
        function burn_step = make_burn_ramp_step(obj, freq_rise_time, ...
                                input_rise_time,burn_amplitude,...
                                hole_freq_offset,wait_freq_offset,output,MEMS,up_not_down)
            name = [num2str(freq_rise_time) 'fqrs,' ...
                    num2str(input_rise_time) 'inrs,' ...
                    num2str(burn_amplitude) 'brnpw,' ...
                    num2str(hole_freq_offset) 'hofs,' ...
                    num2str(wait_freq_offset) 'wofs'];
            freq_rise_samples = freq_rise_time * obj.sample_rate;
            input_rise_samples = input_rise_time * obj.sample_rate;
            amp_wave = [zeros(1,freq_rise_samples-input_rise_samples),...
                    burn_amplitude * (0:1/(input_rise_samples-1):1)];
            freq_wave = flat2flat_transition(freq_rise_samples, ...
                    wait_freq_offset, hole_freq_offset);
            % output gate, sync, and scan marker are both zero for burn pulses
            Os = zeros(size(amp_wave));
            if up_not_down
                burn_step = obj.make_generic_step(['brn-up' name], amp_wave, freq_wave,...
                                output+Os, Os, Os, MEMS+Os);
            else
                burn_step = obj.make_generic_step(['brn-dwn' name], ...
                                fliplr(amp_wave), fliplr(freq_wave), ...
                               output+Os, Os, Os, MEMS+Os);
            end
            burn_step.num_loops = 1;
        end

        % the rise or falling side of the long pulse used to burn a wide spectral
        % hole, or trench.  Frequency is varied sinusoidally while amplitude
        % increases or decreases according to up_not_down
        function burn_step = make_burn_trench_ramp_step(obj, freq_rise_time, ...
                                    input_rise_time,burn_amplitude,...
                                    trench_fraction,up_not_down,odd)
            name = [num2str(freq_rise_time) 'fqrs,' ...
                    num2str(input_rise_time) 'inrs,' ...
                    num2str(burn_amplitude) 'brnpw,' ...
                    num2str(trench_fraction) 'trfr' ...
                    num2str(up_not_down) 'und,' ...
                    num2str(odd) 'odd'];
            freq_rise_samples = freq_rise_time * obj.sample_rate;
            input_rise_samples = input_rise_time * obj.sample_rate;
            if freq_rise_samples < input_rise_samples
                error(['freq_rise_samples (' num2str(freq_rise_samples) ...
                    ') must be greater than input_rise_samples (' ...
                    num2str(input_rise_samples) ')'])
            end
            amp_wave = [burn_amplitude * (0:1/(input_rise_samples-1):1),...
                       burn_amplitude * ones(1,freq_rise_samples-input_rise_samples)];
            freq_wave = flat2flat_transition(freq_rise_samples, ...
                    0, trench_fraction);
            % output gate, sync, and scan marker are both zero for burn pulses
            Os = zeros(size(amp_wave));
            if up_not_down
                burn_step = obj.make_generic_step(['brn-up' name], ...
                                amp_wave, freq_wave, obj.output_block+Os, Os, Os, ...
                                    obj.MEMS_block_output+Os);
            elseif ~up_not_down && ~odd
                burn_step = obj.make_generic_step(['brn-dwn' name], ...
                                fliplr(amp_wave), fliplr(freq_wave), ...
                                obj.output_block+Os, Os, Os, obj.MEMS_block_output+Os);
            elseif ~up_not_down && odd
                burn_step = obj.make_generic_step(['brn-dwn' name], ...
                                fliplr(amp_wave), -fliplr(freq_wave), ...
                                obj.output_block+Os, Os, Os, obj.MEMS_block_output+Os);
            end
            burn_step.num_loops = 1;
        end

        % send a pair of burn pulses; used for acuumulated/Fourier atomic frequency combs
        function burn_steps = make_burn_pair_step_loop(obj, burn_amplitudes, ...
                                    burn_times, wait_times, num_burn_loops, ...
                                    freq_rise_time, input_rise_time, ...
                                    hole_freq_offset, wait_freq_offset)
            % takes up sum(burn_times) + wait_times(1)+wait_times(2) and loops ...
                num_burn_loops times
            name = ['b' num2str(freq_rise_time) 'fqrs' ...
```

```
                            num2str(input_rise_time) 'inrs' ...
                            num2str(burn_times(1)) ',' ...
                            num2str(burn_times(2)) 'tm' ...
                            num2str(burn_amplitudes(1)) ',' ...
                            num2str(burn_amplitudes(2)) 'amp' ...
                            num2str(hole_freq_offset) 'hofs' ...
                            num2str(wait_freq_offset) 'wofs' ...
                            num2str(wait_times(1)) ',' ...
                            num2str(wait_times(2)) 'wt'];

            input_rise_samples = input_rise_time * obj.sample_rate;
            burn_samples = burn_times * obj.sample_rate;
            wait_samples = wait_times * obj.sample_rate;
            if hole_freq_offset == wait_freq_offset
                freq_rise_samples = 0;
            else
                freq_rise_samples = freq_rise_time * obj.sample_rate;
            end

            % wait, transition to fully open when the frequency hits the target
            % burn the first pulse, come back down, wait, rise back up again
            % burn the second pulse, come down again
            amp_wave = [zeros(1,freq_rise_samples-input_rise_samples), ...
                   burn_amplitudes(1) * (0:1/(input_rise_samples-1):1),...
                   burn_amplitudes(1) * ones(1,burn_samples(1)), ...
                   burn_amplitudes(1) * (1:-1/(input_rise_samples-1):0),...
                   zeros(1,wait_samples(1)-2*input_rise_samples),  ...
                   burn_amplitudes(2) * (0:1/(input_rise_samples-1):1), ...
                   burn_amplitudes(2) * ones(1,burn_samples(2)),  ...
                   burn_amplitudes(2) * (1:-1/(input_rise_samples-1):0),...
                   zeros(1,wait_samples(2)-freq_rise_samples-input_rise_samples )];

            if hole_freq_offset == wait_freq_offset
                freq_wave = hole_freq_offset * ones(size(amp_wave));
            else
                freq_wave = [flat2flat_transition(freq_rise_samples, wait_freq_offset, ...
                    hole_freq_offset), ...
                 hole_freq_offset*ones(1,burn_samples(1)+wait_samples(1)+burn_samples(2)), ...
                 flat2flat_transition(freq_rise_samples,hole_freq_offset,wait_freq_offset), ...
                 wait_freq_offset*ones(1,wait_samples(2)-2*freq_rise_samples)];
            end
            % zero for output_marker, scan_marker, sync_marker, MEMS_marker
            Os = zeros(size(amp_wave));
            burn_steps = obj.make_generic_step(name, amp_wave, freq_wave, ...
                        obj.output_block+Os, Os, Os, obj.MEMS_block_output+Os);
            burn_steps.num_loops = num_burn_loops;
        end

        % burn an atomic frequency comb in a piecewise manner (stepping frequency
        % between teeth).
        function burn_steps = make_stepNburn_step_loop(obj, burn_amplitude,...
                                  burn_time, wait_time, num_teeth, num_burn_loops,...
                                  freq_rise_time, input_rise_time, hole_freq_offset,...
                                  teeth_range, wait_freq_offset)
            name = ['b' num2str(num_teeth) 'teeth' ...
                            num2str(teeth_range(1)) ',' ...
                            num2str(teeth_range(2)) 'tr' ...
                            num2str(freq_rise_time) 'fqrs' ...
                            num2str(input_rise_time) 'inrs' ...
                            num2str(burn_time) 'tm' ...
                            num2str(burn_amplitude) 'amp' ...
                            num2str(hole_freq_offset) 'hofs' ...
                            num2str(wait_freq_offset) 'wofs' ...
                            num2str(wait_time) 'wt'];

            if freq_rise_time < wait_time
                error(['freq_rise_time (time from 0 to first freq) must'...
                    'be longer than the first value in wait_time (time'...
                    'to transition frequency between steps)'])
            end

            input_rise_samples = input_rise_time * obj.sample_rate;
            burn_samples = burn_time * obj.sample_rate;
            wait_samples = wait_time * obj.sample_rate;
            freq_rise_samples = freq_rise_time * obj.sample_rate;

            % amplitude waits freq_rise_samples, and then alternates being on for
            % burn_samples and being off for wait_samples, rising with input-rise_samples
            % each time.
            amp_rect = [burn_amplitude * (0:1/(input_rise_samples-1):1),...
                        burn_amplitude * ones(1,burn_samples),...
                        burn_amplitude * (1:-1/(input_rise_samples-1):0),...
                        zeros(1,wait_samples)];
            amp_wave = [zeros(1,freq_rise_samples), repmat(amp_rect, 1, num_teeth),...
                        zeros(1,freq_rise_samples-wait_samples)];

            % frequency rises with freq_rise_samples to the first step, and then
            % alternates waiting burn_samples and stepping in wait_samples to the next
            % value using flat2flat_transition
            if num_teeth > 1
                teeth_freqs = ...
                    (teeth_range(1):range(teeth_range)/(num_teeth-1):teeth_range(2)) + ...
                    hole_freq_offset;
            else
                teeth_freqs = hole_freq_offset;
            end
            freq_steps = num2cell(meshgrid(teeth_freqs, ...
                                1:(burn_samples+2*input_rise_samples)),1);
            freq_moves = arrayfun(@flat2flat_transition, ...
                              [freq_rise_samples, ...
                                  wait_samples*ones(1,num_teeth-1),...
                                  freq_rise_samples],...
                              [wait_freq_offset, teeth_freqs], ...
                              [teeth_freqs, wait_freq_offset], ...
                              'UniformOutput', false);
            freq_wave = freq_moves{1};
            for index = 1:num_teeth % let it grow, LET IT GROW!!!
                freq_wave = [freq_wave freq_steps{index}' freq_moves{index+1} ]; %#ok<AGROW>
            end
            % zero for output_marker, scan_marker, sync_marker, MEMS_marker
            Os = zeros(size(amp_wave));
            burn_steps = obj.make_generic_step(name, amp_wave, freq_wave, ...
                obj.output_block+Os, Os, Os, obj.MEMS_block_output+Os);
            burn_steps.num_loops = num_burn_loops;
        end

        % make a readout step for an AFC (i.e. burn comb once, read out many times).
        % The function makes this one step, while the code to loop the step is
        % commented because repeating any step 40000 times seems to make the AWG ill
        function readout_step = make_readout_step(obj, burn_amplitude,...
                          burn_time, wait_time, freq_offset, output, num_loops)
            % a readout step has the output open
            name = ['read' num2str(burn_time) 'tm' ...
                            num2str(burn_amplitude) 'amp' ...
                            num2str(wait_time) 'wt' ...
                            num2str(freq_offset) 'fqofs' ...
                            num2str(num_loops) 'numloops' ];
            burn_samples = burn_time * obj.sample_rate;
            wait_samples = wait_time * obj.sample_rate;
            amp_wave = burn_amplitude*[ones(1,burn_samples) zeros(1,wait_samples)];
            freq_wave = freq_offset * ones(size(amp_wave));
            scan_marker = amp_wave>0;
            sync_marker = [ones(1,burn_samples) zeros(1,wait_samples)];

            readout_step = obj.make_generic_step(name, ...
                repmat(amp_wave,[1, num_loops]), ...
```

```
                repmat(freq_wave,[1, num_loops]),...
                repmat(output*ones(size(amp_wave)),[1, num_loops]), ...
                repmat(scan_marker,[1, num_loops]),...
                repmat(sync_marker,[1, num_loops]), ...
                repmat(obj.MEMS_pass_output*ones(size(amp_wave)),[1, num_loops])); ...
            readout_step.num_loops = 1;
            % readout_step = obj.make_generic_step(name, amp_wave, freq_wave,...
            %         ones(size(amp_wave)), scan_marker, zeros(size(amp_wave)), ...
            %         ones(size(amp_wave)));
            % readout_step.num_loops = num_loops;
        end

        % makes a step of length obj.trigger_time which triggers the MEMS switch and
        % optionally triggers the sync_marker
        function sync_trigger_step = make_sync_trigger_step(obj, wait_freq_offset)
            name = ['sync_' num2str(obj.trigger_time) 'trigtime,' ...
                          num2str(wait_freq_offset) 'wofs,'];
            trigger_samples = obj.sample_rate * obj.trigger_time;
            amp_wave = zeros(1,trigger_samples);
            freq_wave = wait_freq_offset * ones(size(amp_wave));
            output_wave = ones(size(amp_wave));
            scan_marker = zeros(size(amp_wave));
            sync_marker = sync_to_on * ones(1,trigger_samples);
            MEMS_marker = obj.MEMS_pass_output*ones(1,trigger_samples);
            sync_trigger_step = obj.make_generic_step(name, amp_wave, ...
                         freq_wave, output_wave, scan_marker, sync_marker, MEMS_marker);
            sync_trigger_step.num_loops = 1;
        end

        % makes a step that includes a square on the sync marker and a frequency scan
        function scan_step = make_scan_n_triggersync_step(obj, freq_rise_time, ...
            input_rise_time, ...
                                       scan_time, scan_freq_range, scan_amplitude, ...
                                           wait_freq_offset)
            name = ['scn' num2str(freq_rise_time) 'fqrs,' ...
                          num2str(input_rise_time) 'inrs,' ...
                          num2str(scan_time) 'tm,' ...
                          num2str(scan_freq_range) 'scnrng,'...
                          num2str(scan_amplitude) 'scnpw'...
                          num2str(wait_freq_offset) 'wofs'];
            freq_rise_samples = freq_rise_time * obj.sample_rate;
            input_rise_samples = input_rise_time * obj.sample_rate;
            scan_samples = scan_time * obj.sample_rate;
            amp_wave = [zeros(1,freq_rise_samples-input_rise_samples) ...
                    scan_amplitude * (0:1/(input_rise_samples-1):1),...
                    scan_amplitude * ones(1,scan_samples),...
                    scan_amplitude * (1:-1/(input_rise_samples-1):0)...
                    zeros(1,freq_rise_samples-input_rise_samples)];
            freq_wave = smooth_transition_scan(2*freq_rise_samples...
                    +scan_samples, scan_samples, scan_freq_range,wait_freq_offset);
            output_wave = obj.output_pass*ones(size(amp_wave));
            scan_marker = [ones(1,floor(length(amp_wave)/2)) ...
                           zeros(1,ceil(length(amp_wave)/2)) ];
            trigger_samples = obj.trigger_time * obj.sample_rate;
            sync_marker = [ones(1,trigger_samples), ...
                zeros(1,length(scan_marker)-trigger_samples)];
            MEMS_marker = obj.MEMS_pass_output*ones(size(amp_wave));
            scan_step = obj.make_generic_step(name, amp_wave, freq_wave, output_wave, ...
                                          scan_marker, sync_marker, MEMS_marker);
            scan_step.num_loops = 1;
        end

        % breaks a single wait step into a looped shorter step and a remainder step
        function wait_steps = frac_n_make_wait_step(obj, flat_time, freq_offset, output, MEMS)
            if flat_time <= 0 % so that it doesn't die for wait_time == rise_time/2
                wait_steps = [];
            else
                % split the wait time into blocks of length obj.max_flat_time
                num_loops = floor(flat_time / obj.max_flat_time);
                remainder = mod(flat_time, obj.max_flat_time);

                if num_loops > 0
                    wait_loop = obj.make_flat_step(obj.max_flat_time, 0, ...
                        freq_offset,output,MEMS);
                    wait_loop.num_loops = num_loops;
                end

                if remainder > 0        % don't make an empty wait step
                    wait_remainder = obj.make_flat_step(round(remainder,10), 0, ...
                        freq_offset,output,MEMS);
                    wait_remainder.num_loops = 1;
                    if num_loops > 0
                        wait_steps = [wait_remainder wait_loop];
                    else
                        wait_steps = wait_remainder;
                    end
                else
                    wait_steps = wait_loop;
                end
            end
        end

        % a flat step is assumed to have 1 in the mems marker and 0 in the sync marker,
        % thus triggering neither.  To actually trigger, use make_readout_step
        function flat_step = make_flat_step(obj, flat_time, amplitude, freq_offset,output,MEMS)
            name = ['flat' num2str(flat_time) ',' ...
                        num2str(amplitude) 'amp,' ...
                        num2str(freq_offset) 'frqofst'];
            % flat_samples = round(flat_time) * obj.sample_rate;  %%%%%%% pretty sure this ...
                was wrong
            flat_samples = round(flat_time * obj.sample_rate);
            amp_wave = amplitude * ones(1,flat_samples);
            output_wave = output * ones(size(amp_wave));
            freq_wave = freq_offset * ones(1,flat_samples);
            scan_marker = zeros(size(amp_wave));
            sync_marker = zeros(size(amp_wave));
            MEMS_marker = MEMS*ones(size(amp_wave));
            flat_step = obj.make_generic_step(name, amp_wave, freq_wave, output_wave, ...
                                              scan_marker, sync_marker, MEMS_marker);
        end

        % a flat step is assumed to have 1 in the mems marker and 0 in the sync marker,
        % thus triggering neither.  To actually trigger, use make_readout_step
        function cosine_step = ...
            make_cosine_step(obj,cosine_time,freq_mod_period,freq_amplitude,amplitude)
            name = ['cos' num2str(cosine_time) ',' ...
                        num2str(freq_amplitude) 'trfr,' ...
                        num2str(amplitude) 'amp'];
            % flat_samples = round(flat_time) * obj.sample_rate;  %%%%%%% pretty sure this ...
                was wrong
            cosine_samples = cosine_time * obj.sample_rate;
            freq_mod_period_samples = freq_mod_period * obj.sample_rate;
            amp_wave = amplitude * ones(1,cosine_samples);
            output_wave = obj.output_block * ones(size(amp_wave));
            freq_wave = freq_amplitude * cos(2*pi*(1:cosine_samples)/freq_mod_period_samples);
            scan_marker = zeros(size(amp_wave));
            sync_marker = zeros(size(amp_wave));
            MEMS_marker = obj.MEMS_block_output*ones(size(amp_wave));
            cosine_step = obj.make_generic_step(name, amp_wave, freq_wave, output_wave, ...
                                              scan_marker, sync_marker, MEMS_marker);
        end

        % makes a step struct and, if it doesn't already exist, uploads it to the AWG
        function step = make_generic_step(obj, name, amp, freq, output, scan_marker, ...
            sync_marker, MEMS_marker)
            name = strrep(name, '0.', '.');
            if length(name)>=64
                error(['step name "' name '" is too long.  ruh roh...);']);
            end
            if ~obj.is_name_loaded(name)
                step.name = name;
                step.amp_wave = amp;
                step.freq_wave = freq;
                step.input_marker = amp>0;      % shows when input is on
                step.sync_marker = sync_marker;
                step.scan_marker = scan_marker;
                step.MEMS_marker = MEMS_marker;
                step.output_wave = output;
                step.output_marker = output>0;  % shows when output_wave is on
                obj.upload_step(step)
            else
                step = obj.get_step_from_name(name);
            end
        end

%% utility commands

        % determines if a step is loaded using the cell array of loaded steps
        function loaded = is_name_loaded(obj, name)
            loaded = false;
            for loadedstep = obj.loaded_steps
                if ~isempty(loadedstep)
                    loaded = loaded || strcmp(name, loadedstep{1}.name);
                end
            end
        end

        % finds and returns the full step object for a given step name
        % using the cell array of loaded steps obj.loaded_steps
        function step = get_step_from_name(obj, step_name)
            for loadedstep = obj.loaded_steps
                if strcmp(loadedstep{1}.name, step_name)
                    step = loadedstep{1};
                    break
                end
            end
        end

        % loads the frequency and amplitude components into the AWG also
        % chooses which markers to load for each step currently set to
        % upload input_marker to marker2 and either scan or output
        % marker to amp&freq or output waves for marker 1, respectively
        function upload_step(obj, step)
            if length(step.amp_wave)~=length(step.input_marker) || ...
               length(step.amp_wave)~=length(step.sync_marker) || ...
               length(step.amp_wave)~=length(step.scan_marker) || ...
               length(step.amp_wave)~=length(step.MEMS_marker)
                error(['Size mismatch on generated waveforms for ' ...
                       step.name '  If you reached this using a '...
                       'Sequence_loader function, you''ve earned a '...
                       'phone-a-friend: 949-370-0707.'])
            end

            obj.loaded_steps{length(obj.loaded_steps)+1} = step;
            obj.Awg_instance.create_waveform(strcat('a_',step.name),...
                            step.amp_wave, step.input_marker, step.sync_marker);
            obj.Awg_instance.create_waveform(strcat('f_',step.name),...
                            step.freq_wave, step.scan_marker, step.MEMS_marker);
            if obj.output_channel_on
                obj.Awg_instance.create_waveform(strcat('o_',step.name),...
                            step.output_wave, step.output_marker, step.input_marker);
            end
            obj.total_loaded_points = obj.total_loaded_points + ...
                                     length(step.amp_wave)*(2+obj.output_channel_on);
            if obj.verbose
                if obj.output_channel_on
                    disp(['   uploaded amp,freq,&output for step: ' step.name]);
                else
                    disp(['   uploaded amp&freq for step: ' step.name]);
                end
            end
        end

        % Adds steps by name to the end of the sequence.  Assumes names are
        % unique identifiers, and waveforms by those names have already been
        % uploaded to the AWG
        function append_to_sequence(obj, steps)
            for step = steps
                step_num = obj.Awg_instance.get_sequence_num_steps()+1;
                obj.Awg_instance.set_sequence_num_steps(step_num);
                obj.Awg_instance.set_channel_waveform_seq_step(obj.input_channel_num, ...
                    step_num,['a_' step.name]);
                obj.Awg_instance.set_channel_waveform_seq_step(obj.freq_channel_num, ...
                    step_num,['f_' step.name]);
                if obj.output_channel_on
                    obj.Awg_instance.set_channel_waveform_seq_step(obj.output_channel_num, ...
                        step_num,['o_' step.name]);
                end
                assert(isfield(step,'num_loops'))
                if step.num_loops ~= 1
                    obj.Awg_instance.set_channel_seq_step_loop_num(step_num, step.num_loops)
                end
                if obj.verbose
                    disp(['   appended step: ' step.name ' (loop:' num2str(step.num_loops) ')']);
                end
                pause(.5)
            end
        end

        % causes the sequence to loop back to the first step after completing the last
        function close_sequence_loop(obj)
            num_steps = obj.Awg_instance.get_sequence_num_steps();
            obj.Awg_instance.set_sequence_step_goto(num_steps,1);
        end

        % plot the loaded steps. A downsample factor of ~100 is recommended,
        % as the number of total points will likely be large.
        function plot_handle = plot_current(obj,downsample_factor)
            channel_a_waveform = [];             channel_b_waveform = [];
            sync_marker_waveform = [];           input_marker_waveform = [];
            scan_marker_waveform = [];           MEMS_marker_waveform = [];
            channel_c_waveform = [];             output_marker_waveform = [];

            for step = obj.current_steps
                channel_a_waveform = [channel_a_waveform downsample(repmat(step.amp_wave, ...
                                      [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                channel_b_waveform = [channel_b_waveform downsample(repmat(step.freq_wave, ...
                                      [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                input_marker_waveform = [input_marker_waveform ...
                    downsample(repmat(step.input_marker, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                sync_marker_waveform = [sync_marker_waveform ...
                    downsample(repmat(step.sync_marker, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                scan_marker_waveform = [scan_marker_waveform ...
                    downsample(repmat(step.scan_marker, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                MEMS_marker_waveform = [MEMS_marker_waveform ...
                    downsample(repmat(step.MEMS_marker, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                if obj.output_channel_on
```

```
                    channel_c_waveform = [channel_c_waveform ...
                        downsample(repmat(step.output_wave, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                    output_marker_waveform = [output_marker_waveform ...
                        downsample(repmat(step.output_marker, ...
                                    [1, step.num_loops]),downsample_factor)]; %#ok<AGROW>
                end
            end
            plot_handle = figure();
            total_samples = length(scan_marker_waveform);
            x = (1:total_samples)/obj.sample_rate/1000*downsample_factor;
            plot(x, channel_a_waveform, x, channel_b_waveform, x, input_marker_waveform, x, ...
                sync_marker_waveform, ...
                 x, scan_marker_waveform, x, MEMS_marker_waveform);
            xlabel('seconds')
            ylabel('volts')
            if obj.output_channel_on
                hold on
                plot(x, channel_c_waveform, x, output_marker_waveform)
                hold off
                legend('input open', 'frequency offset', 'RF switch', 'sync marker', 'scan ...
                    trigger', ...
                    'MEMS trigger', 'output wave','Location','SouthEast');
            else
                legend('input open', 'frequency offset', 'RF switch', 'sync marker', 'scan ...
                    trigger', ...
                    'Location','SouthEast');
            end
        end

        % used to determine if rounding of samples due to division of total
        % time by sample rate will lead to rounding errors.  If a warning is
        % given, check that you're tolerant of the amount rounded.
        function check_rounding(obj, time_vector)
            for time = time_vector
                num_samples = time * obj.sample_rate;
                if mod(num_samples,1)~=0
                    warning(['may be rounding value ' num2str(time) ' ms by ' ...
                        num2str(time - round(num_samples)/obj.sample_rate) ' ms'])
                end
            end
        end
    end
end
```

B.3 Tunics TECL

```
% Tunics_TECL_obj.m
%
% Interface object to control a TUNICS PLUS SC External Cavity Diode Laser
% Built starting from lfm.m and the instrument control toolbox.  Note that
% frequency commands are not completely implemented.
%
% more information on commands used in
% www.equipland.com/objects/catalog/product/extras/1520_Photonetics_Tunics_PR_PRI_Manual.pdf
%
% usage:
%     laser_instance = Tunics_TECL_obj('GPIB1::2::0::INSTR',0.5);
%     laser_instance.lase();
%     laser_instance.set_wavelength_nm(1536.45);
%     power = laser_instance.get_power_mW();
%     laser_instance.lasing_off();
%     laser_instance.close();
%
```

```
% ETM 20150407

classdef (ConstructOnLoad = true) Tunics_TECL_obj < handle
    properties (SetAccess = private)
        Tunics_visa;
        pause_time=0.5; % pause time after sending commands
    end
    methods
        function obj = Tunics_TECL_obj(address, time_to_pause)
            if nargin >= 1 % use the given address, if provided
                if ~ischar(address)
                    address = 'GPIB1::2::0::INSTR';
                    disp([ 'invalid address, using ' address ])
                end
            end
            instruments = instrfind('Type', 'visa-gpib', 'RsrcName', address, 'Tag', '');

            if isempty(instruments) % Create the VISA-GPIB object if it does not exist
                obj.Tunics_visa = visa('AGILENT', address);
            else            % otherwise use the object that was found.
                fclose(instruments);
                obj.Tunics_visa = instruments(1);
            end

            fopen(obj.Tunics_visa);     % Connect to instrument object, obj.

            if nargin == 2 % use the given wait time, if provided
                obj.pause_time = max(0.5,time_to_pause);
            end
        end
        function lase(obj)
            fprintf(obj.Tunics_visa, 'ENABLE');
            pause(obj.pause_time);
        end
        function lasing_off(obj)
            fprintf(obj.Tunics_visa, 'DISABLE');
            pause(obj.pause_time);
        end
      function close(obj)
            fclose(obj.Tunics_visa);    % Disconnect all objects.
        end
        % function start_sweep(obj)
        %     fprintf(obj.Tunics_visa, 'SCAN');
        % end       Useless: SCAN doesn't work on this laser for some reason
        function stop_sweep(obj)
            fprintf(obj.Tunics_visa, 'STOP');
        end
        %% getters
        function current = get_current_mA(obj)
            query(obj.Tunics_visa, 'I?');
            current = fscanf(obj.Tunics_visa);
        end
        function power = get_power_mW(obj)
            query(obj.Tunics_visa, 'P?');
            power = fscanf(obj.Tunics_visa);
        end
        function wavelength = get_wavelength_nm(obj)
            query(obj.Tunics_visa, 'L?');
            wavelength = fscanf(obj.Tunics_visa);
        end
        function freq = get_freq_GHz(obj)
            query(obj.Tunics_visa, 'f?');
            freq = fscanf(obj.Tunics_visa);
        end
        function is_lim = is_at_limit(obj)
            query(obj.Tunics_visa, 'LIMIT?');
            is_lim = fscanf(obj.Tunics_visa);
        end
        %% setters
        function set_const_power(obj) % varies current to reduce power noise
            fprintf(obj.Tunics_visa, 'APCON');
            pause(obj.pause_time);
        end
        function set_const_current(obj) % sets constant current
            fprintf(obj.Tunics_visa, 'APCOFF');
            pause(obj.pause_time);
        end
        function set_fine_scan(obj, delta_lambda)
            fprintf(obj.Tunics_visa, ['FSCL=' num2str(delta_lambda, '%03.1f')]);
        end
        % function configure_scan_range_nm(obj, min_wavelength, delta_lambda, max_wavelength)
        %     fprintf(obj.Tunics_visa, ['Smin=' num2str(min_wavelength, '%08.3f')]);
        %     fprintf(obj.Tunics_visa, ['Smax=' num2str(delta_lambda, '%08.3f')]);
        %     fprintf(obj.Tunics_visa, ['Step=' num2str(max_wavelength, '%04.3f')]);
        % end       Useless: SCAN doesn't work on this laser for some reason
        function set_scan_dwell_sec(obj, dwell_time)
            fprintf(obj.Tunics_visa, ['Stime=' num2str(dwell_time, '%03.1f')]);
        end
        function set_power_mW(obj,power)
            fprintf(obj.Tunics_visa, ['P=' num2str(power, '%05.2f')]);
            pause(obj.pause_time);
        end
        function set_current_mA(obj, current)
            fprintf(obj.Tunics_visa, ['I=' num2str(current, '%04.1f')]);
        end
        function set_wavelength_nm(obj, wavelength)
            if wavelength <1450 || wavelength>1590
                error('wavelength out of range')
            else
                fprintf(obj.Tunics_visa, ['L=' num2str(wavelength, '%08.3f')]);
            end
            pause(obj.pause_time);
        end
        function set_frequency_GHz(obj, frequency)
            fprintf(obj.Tunics_visa, ['I=' num2str(frequency, '%08.1f')]);
        end
    end
end
```

B.4 Toptica DLC pro

```
% Toptica_DLCpro_obj.m
%
% Interface object to control a Toptica CTL and DLC Pro
% Built from Tunics_TECL_obj.m and the matlab instrument control toolbox
%
% DLCpro command reference on USB stick with laser manuals in the filecabinet
% copied to ~\Documents\Toptica documentation\1_TOPTICA DLC pro
%     SOFTWARE_1.3.1\1_DOCUMENTATION\Toptica_DLCpro-Command-Reference
%
% Most errors that arise seem to be caused by the read buffer not being
% cleared properly.  In the event of something weird, try uncommenting
% that either the end of the constructor or changing set_n_confirm()
%
% usage:
%     laser_instance = Toptica_DLCpro_obj('131.215.48.207');
%                    % or: Toptica_DLCpro_obj('COM5');
%     laser_instance.lase();
%     laser_instance.set_wavelength_nm(1536.45);
%     power = laser_instance.get_power_mW();
%     laser_instance.lasing_off();
%     laser_instance.close();
```

```
%
% ETM 20151016

classdef (ConstructOnLoad = true) Toptica_DLCpro_obj < handle
    properties (SetAccess = private)
        DLCpro_visa;
        bool_str = {'#f';'#t'}
    end
    methods
        function obj = Toptica_DLCpro_obj(address_str)
            % Connect via USB
            if strncmpi(address_str, 'COM',3) || strncmpi(address_str, 'USB',3)
                instruments = instrfind('Type', 'serial', 'Port', address_str, 'Tag', '');

                if isempty(instruments) % Create the VISA-serial object if it does not exist
                    obj.DLCpro_visa = serial(address_str);
                else            % otherwise use the object that was found.
                    fclose(instruments);
                    obj.DLCpro_visa = instruments(1);
                end
            % Connect via ethernet
            else % check naively that address_str is IP-like (doesn't check <255)
                if regexp(address_str,'^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$')
                    instr_address = address_str;
                else
                    instr_address  = '131.215.48.207'; % last I checked, this was it
                    warning(['given address invalid, using ' instr_address]);
                end
                obj.DLCpro_visa = tcpip(instr_address, 1998);
            end
            fopen(obj.DLCpro_visa);     % Connect to instrument object, obj.

            fprintf(obj.DLCpro_visa, '(param-set! ''echo #f)');
            flushinput(obj.DLCpro_visa);
            flushoutput(obj.DLCpro_visa);

            % % debug code in case reading information fails
            % echo = fscanf(obj.DLCpro_visa);
            % if echo ~= '0'
            %     disp('echo off is acting funny');
            %     if strcmp(echo,'(param-set! ''echo #f)') == 1
            %         disp('woah, boy.  acting real funny-like');
            %         echo = fscanf(obj.DLCpro_visa);
            %         if echo ~= '0'
            %             error(['something went sideways.  Should print 0 but got ' echo])
            %         end
            %     end
            %     fscanf(obj.DLCpro_visa);
            % end

            % have some fun
            fprintf(obj.DLCpro_visa, '(exec ''buzzer:welcome "EEEE    EEEE    EEEE    AAA  ...
                HH  EEEE    AAA  HH  EEEEEE      KKKK    KKKK    KKKK   LLL  GG  CCCC    AAA ...
                  GG  EEEEEE ")');
        end
      function close(obj)
            fclose(obj.DLCpro_visa);    % Disconnect all objects.
        end
        function start_sweep(obj)       % sweep using coarse (motor) scan
         fprintf(obj.DLCpro_visa, '(exec ''laser1:ctl:scan:start)');
        end
        function stop_sweep(obj)
         fprintf(obj.DLCpro_visa, '(exec ''laser1:ctl:scan:stop)');
        end
        function pause_sweep(obj)
         fprintf(obj.DLCpro_visa, '(exec ''laser1:ctl:scan:pause)');
        end
        function resume_sweep(obj) % can only resume a paused sweep
```

```
            fprintf(obj.DLCpro_visa, '(exec ''laser1:ctl:scan:continue)');
        end
        function sing(obj,song_num) % you're welcome
            if(song_num==1)
                fprintf(obj.DLCpro_visa, '(exec ''buzzer:play "AAAA   AA  A  AAAA  CC  B  BB ...
                     A  AAA  A  AAAAAA")');
            else
                fprintf(obj.DLCpro_visa, '(exec ''buzzer:play "EEEE    EEEE    EEEE    AAA  ...
                    HH  EEEE    AAA  HH  EEEEEE      KKKK    KKKK    KKKK   LLL  GG  CCCC    ...
                    AAA   GG  EEEEEE ")');
            end
        end
        %% getters
        function power = get_power_mW(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:power:power-act)');
            output = fscanf(obj.DLCpro_visa);
            power = str2double(output(3:end-2));
        end
        function current = get_current_mA(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:dl:cc:current-act)');
            output = fscanf(obj.DLCpro_visa);
            current = str2double(output(3:end-2));
        end
        function wavelength = get_wavelength_nm(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:wavelength-act)');
            output = fscanf(obj.DLCpro_visa);
            wavelength = str2double(output(3:end-2));
        end
        function voltage = get_piezo_actual_voltage(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:dl:pc:voltage-act)');
            output = fscanf(obj.DLCpro_visa);
            voltage = str2double(output(3:end-2));
        end
        function voltage = get_piezo_offset_voltage(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:scan:offset)');
            output = fscanf(obj.DLCpro_visa);
            voltage = str2double(output(3:end-1));
        end
        function sweep_bounds = get_motor_sweep_bounds(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:scan:wavelength-begin)');
            sweep_start = fscanf(obj.DLCpro_visa);
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:scan:wavelength-end)');
            sweep_end = fscanf(obj.DLCpro_visa);
            sweep_bounds = [str2double(sweep_start(3:end-2)) str2double(sweep_end(3:end-2))];
        end
        function scale_factor = get_piezo_scaling_factor(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:dl:pc:external-input:factor)');
            string = fscanf(obj.DLCpro_visa);
            scale_factor = str2double(string(3:end));
        end
        % returns if the system should tune the piezo with external input
        % getting the right channel and a nontrivial tuning range
        function enabled = get_external_piezo_enabled(obj)
            external_enabled = query(obj.DLCpro_visa, '(param-ref ...
                ''laser1:dl:pc:external-input:enabled)');
            input_factor =  query(obj.DLCpro_visa, '(param-ref ...
                ''laser1:dl:pc:external-input:factor)');
            input_channel = query(obj.DLCpro_visa, '(param-ref ...
                ''laser1:dl:pc:external-input:signal)');
            enabled = strcmp(external_enabled(3:4), '#t') && ...
                (str2double(input_factor(3:end))>0) && ...
                isempty(find([0,1,2,4]==input_channel,1));
        end
        % accurately tune to a given frequency using a wavemeter
        % needs debugging because the wavemeter failed before I finished
        % function goto_freq_closedloop(obj, goto_freq_GHz, wavemeter_obj)
        %     c = 299792458;
        %     obj.set_wavelength_nm(c/goto_freq_GHz);

        %     volts_per_GHz = 10/2.1;  % going from 69V to 79V moved 2.1 GHz
        %     actual_freq = wavemeter_obj.get_freq_GHz();
        %     pause(1);
        %     while abs(actual_freq - goto_freq_GHz) > 0.05
        %         % dlambda = -c / f^2 * df
        %         obj.set_piezo_voltage( volts_per_GHz * c / goto_freq_GHz^2 *...
        %                               (freq-wavemeter.get_freq_GHz()) ...
        %                               + obj.get_piezo_actual_voltage() );
        %         pause(1);
        %         actual_freq = wavemeter_obj.get_freq_GHz();
        %         pause(1);
        %     end
        % end

        %% setters
        function set_const_power(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:power-stabilization:enabled #t)');
        end
        function set_const_current(obj)
            fprintf(obj.DLCpro_visa, '(param-ref ''laser1:ctl:power-stabilization:enabled #f)');
        end
        function configure_scan_range_nm(obj, min_wavelength, max_wavelength, speed, loop_sweep)
            set_n_confirm(obj, ['(param-set! ''laser1:ctl:scan:wavelength-begin' ...
                num2str(min_wavelength, '%08.3f') ')']);
            set_n_confirm(obj, ['(param-set! ''laser1:ctl:scan:wavelength-end' ...
                num2str(max_wavelength, '%08.3f') ')']);
            set_n_confirm(obj, ['(param-set! ''laser1:ctl:scan:speed' num2str(speed, ...
                '%08.3f') ')']);
            set_n_confirm(obj, ['(param-set! ''laser1:ctl:scan:continuous-mode' ...
                obj.bool_str{1+loop_sweep} ')']);
            set_n_confirm(obj, '(param-set! ''laser1:ctl:scan:microsteps #t)');
        end
        function set_power_mW(obj,power)
            set_n_confirm(obj, ['(param-set! ''laser1:ctl:power-stabilization:setpoint ' ...
                num2str(power, '%05.2f') ')']);
        end
        function set_current_mA(obj, current)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:cc:current-set ' num2str(current, ...
                '%04.1f') ')' ]);
        end
        function set_wavelength_nm(obj, wavelength)
            % could read the limits from the laser, but I hardcoded it because I'm lazy
            if wavelength <1460 || wavelength>1570
                error('wavelength out of range')
            else
                set_n_confirm(obj, ['(param-set! ''laser1:ctl:wavelength-set ' ...
                    num2str(wavelength) ')']);
            end
        end
        function set_piezo_enable(obj, true_for_on)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:enabled ' ...
                obj.bool_str{1+true_for_on} ')']);
        end
        function set_piezo_voltage(obj, voltage)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:voltage-set ' num2str(voltage) ')']);
        end
        function set_piezo_dithering(obj, true_for_on)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:voltage-set-dithering ' ...
                obj.bool_str{1+true_for_on} ')']);
        end
        function set_piezo_external_channel(obj, front_channel_num)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:external-input:signal ' ...
                num2str(front_channel_num) ')']);
        end
        function set_piezo_scaling_factor(obj, scaling_factor)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:external-input:factor ' ...
                num2str(scaling_factor) ')']);
        end
        function set_piezo_external_control(obj, enable)
            set_n_confirm(obj, ['(param-set! ''laser1:dl:pc:external-input:enabled ' ...
                obj.bool_str{1+enable} ')']);
        end

        % internal set command that confirms
        function set_n_confirm(obj,command_string)
            value = query(obj.DLCpro_visa, command_string);
            if(value ~= '0')
                error(['something didn''t set correctly.  instead got value: ' value])
            end
        end

    end
end
```

B.5 Tektronix Oscilloscope

```
% Tektronix_TDS2014B.m
%
% Interface object to control a Tektronix TDS2014B oscilloscope
% Built from a few sources
%   http://www1.tek.com/forum/viewtopic.php?f=6&t=3217
%   AWG2014 programmer manual from online saved in fgen_libraries
%   Jon's implementation of the 2024c Tex mdd file readscope
%
% does not implement possible trigger commands
%         % to get list of settings that can be configured
%         set(osc)
%         % to get the current configuration of the oscilloscope
%         get(osc)
%         % configuring trigger (as example of setting up scope)
%         trigger_group = get(osc, 'Trigger');
%         % use get command to read property
%         get(trigger_group, 'Slope' );
%         % use set command to set.
%         set(trigger_group, 'Slope', 'falling');
%         % can get list of other properties using
%         get(trigger_group)
%
% usage:
%     scope_instance = Tektronix_AWG5014('USB0::0x0699::0x0368::C034313::0::INSTR');
%     scope_instance.set_channels_on([1,1,0,1]);
%     scope_instance.set_num_averaging(2);
%     scope_instance.get_waveform(2); % channel 2
%     scope_instance.close(2);
%
% ETM 20151130

classdef (ConstructOnLoad = true) Tektronix_TDS2014B < handle
    properties (SetAccess = private)
        tds_obj;
        buffer_size;
        acq_obj;
        waveform_group;
        chan_obj;
        num_values = 2500;
        channels_on;
    end
    methods
        function obj = Tektronix_TDS2014B(address)

```

```
            if ischar(address)
                instr_address = address;
            else
                instr_address = 'USB0::0x0699::0x0368::C034313::0::INSTR';
            end

            % Find a VISA-USB object.
            instr_visa = instrfind('Type', 'visa-usb', 'RsrcName', instr_address, 'Tag', '');

            % % in case it doesn't work, try this
            % disconnect(obj.tds_obj)
            % delete(obj.tds_obj)

            % Create the VISA-USB object if it does not exist
            % otherwise use the object that was found.
            if isempty(instr_visa)
                instr_visa = visa('AGILENT', instr_address);
            else
                fclose(instr_visa);
                instr_visa = instr_visa(1);
            end

            obj.tds_obj = icdevice('tektronix_tds2014.mdd', instr_visa);

            % Connect device object to hardware.
            connect(obj.tds_obj);

            obj.acq_obj = get(obj.tds_obj,'Acquisition');
            wvfm_group = get(obj.tds_obj, 'Waveform');
            obj.waveform_group = wvfm_group(1);
            obj.chan_obj = get(obj.tds_obj,'Channel');

            for channel = 1:4
                obj.channels_on(channel) = strcmp(obj.chan_obj.State(channel),'on');
            end
        end

        %% control
      function close(obj)
            disconnect(obj.tds_obj);
            delete(obj.tds_obj);
        end
        % check every 100 ms to make sure the scope is not busy before
        % continuing.  Breaks after max_time_sec
        function finish_last_command(obj, max_time_sec)
            warning('this command doesn''t seem to work')
            for i=1:max_time_sec/10;
                if ~obj.tds_obj.Busy
                    return;
                end
                pause(10)
            end
            warning('command did not finish in allotted time')
        end
        %% getters
        function acq_settings = get_acquisition_settings(obj)
            acq_settings = obj.acq_obj;
        end
        function chan_settings = get_channel_settings(obj)
            chan_settings = obj.chan_obj;
        end
        function [X, Y] = get_waveform(obj,channel_nums)
            if ~isempty(channel_nums)
                if isscalar(channel_nums)
                    [Y, X] = invoke(obj.waveform_group, ...
                            'readwaveform', ['channel' num2str(channel_nums)]);
                else
                    if max(channel_nums)>4 || min(channel_nums)<1
```

```
                            error('oh come on... channel numbers are 1-4')
                        end
                        Y = zeros(obj.num_values,4);
                        X = zeros(obj.num_values,4);
                        for channel_num = channel_nums
                            disp(['sending in ->channel' num2str(channel_num) '<-'])
                            [y, x] = invoke(obj.waveform_group, ...
                                'readwaveform', ['channel' num2str(channel_num)]);
                            Y(:,channel_num) = y;
                            X(:,channel_num) = x;
                        end
                    end
                end
            end
            function channel_bits = get_channels_on(obj)
                channel_bits = obj.channels_on;
            end
            %% setters
            function set_single_acquisition(obj)
                set(obj.acq_obj,'State','run')
                set(obj.acq_obj,'Control','single')
            end
            % number of frames to average
            function set_num_averaging(obj, num_frames)
                set(obj.acq_obj,'NumberOfAverages',num_frames)
            end
            function set_timebase_seconds(obj, time)
                    set(obj.acq_obj,'Timebase',time)
            end
            % delay between trigger and acquisition window
            function set_timedelay_seconds(obj, time)
                    set(obj.acq_obj,'Delay',time)
            end
            % give a 'binary' vector to turn channels on/off
            function set_channels_on(obj, channel_bits)
                if length(channel_bits)==4
                    for channel_num = 1:4
                        if logical(channel_bits(channel_num))
                            set(obj.chan_obj(channel_num),'State','on');
                        else
                            set(obj.chan_obj(channel_num),'State','off');
                        end
                    end
                    obj.channels_on = channel_bits;
                else
                    error('channel_bits must be 1x4, setting all channels');
                end
            end
            function set_voltage_scale(obj, V, channel_nums)
                for channel_num = channel_nums
                    set(obj.chan_obj(channel_num),'Scale',V);
                end
            end
            function set_voltage_position(obj, V, channel_nums)
                for channel_num = channel_nums
                    set(obj.chan_obj(channel_num),'Position',V);
                end
            end

        end
    end
```