DYNAMICS OF TRAPPED POLARITONS IN STRESSED GaAs QUANTUM WELL-MICROCAVITY STRUCTURES: EXPERIMENTS AND NUMERICAL SIMULATIONS

by

Vincent Edward Hartwell

B.S., Univeristy of Texas at Austin, 1990

M.S., University of South Florida, 1995

Submitted to the Graduate Faculty of

the Department of Physics and Astronomy in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF PHYSICS AND ASTRONOMY

This dissertation was presented

by

Vincent Edward Hartwell

It was defended on

September 3, 2008

and approved by

Dr. David Snoke, Physics and Astronomy

Dr. Robert Coalson, Chemistry

Dr. Kevin Chen, Electrical Engineering

Dr. Rainer Johnsen, Physics and Astronomy

Dr. Robert P. Devaty, Physics and Astronomy

Dissertation Director: Dr. David Snoke, Physics and Astronomy

DYNAMICS OF TRAPPED POLARITONS IN STRESSED GaAs QUANTUM WELL-MICROCAVITY STRUCTURES: EXPERIMENTS AND NUMERICAL SIMULATIONS

Vincent Edward Hartwell, PhD

University of Pittsburgh, 2008

Microcavity polaritons have been studied for a decade and a half. Soon after their discovery they were proposed as candidates for the observation of BEC in a solid. In consideration of this possibility, microcavity polaritons have been studied experimentally, analytically, and numerically. Most of the numerical studies have been qualitative. This thesis continues that analysis and for the first time fits experimentally obtained distributions with that obtained by numerical simulations.

For this thesis, experiments were performed on a GaAs quantum well-microcavity structure. Excitations of this structure are manifested as polaritons when the quantum well excitons are strongly coupled to the cavity mode. The experimental study of these polaritons provides interesting results. The experiments where the polariton density is the highest show that there is accumulation of polaritons in the low energy states near $k = 0$. Below this high density it is seen that the distribution becomes flat and maintains that shape as density is decreased. Neither the high density nor the low density data has a thermalized distribution. Can the accumulation at high density be explained with Boson statistics? What can explain the flat, nonthermalized distribution at low densities. To answer these questions a numerical model was developed. The model has shown that the distribuition functions from the experiments can be numerically simulated. The model has shown that the accumulation at $k = 0$ is due to Boson statistics. Through the model, an explanation as to why the distribution curves are flat is also provided.

This thesis is presented as follows. An introduction to microcavity polaritons and to our experimental system is presented in chapter 1. Chapter 2 describes the scattering processes that regulate the dynamics of the polaritons and the equations that are used in the model. Chapter 3 gives a review of previous numerical models on microcavity polaritons. Chapter 4 describes the experimental techniques used to acquire the data while chapter 5 compares the data with that given by the simulation. Chapter 6 then discusses directions for continued research.

TABLE OF CONTENTS

PREFACE xvi

1.0 INTRODUCTION 1

1.1 CAVITY MODES 2

1.2 QUANTUM WELL EXCITONS 4

1.3 POLARITONS 7

1.4 POLARITON TRAPPING 15

1.5 EXPERIMENTAL SETUPS 18

1.5.1 MIRA 20

1.5.2 AO MODULATION 22

1.5.3 CRYOSTAT AND STRESSOR INSERT 22

1.5.4 IMAGING SPECTROMETER AND CCD CAMERA 24

2.0 THE MODEL 26

2.1 SCATTERING 26

2.1.1 POLARITON-POLARITON INTERACTIONS 26

2.1.2 POLARITON-LONGITUDINAL ACOUSTICAL PHONON INTERACTIONS 31

2.1.3 POLARITON-TRANSVERSE ACOUSTICAL PHONON INTERACTIONS 34

2.1.4 POLARITON-PHONON INTERACTION BY PIEZOELECTRICTY 34

2.1.5 POLARITON-OPTICAL PHONON INTERACTIONS 36

2.1.6 FREE ELECTRON-POLARITON INTERACTIONS 37

2.2 ENERGY CORRECTIONS 39

2.2.1 FIRST-ORDER ENERGY CORRECTION . . . 39
2.2.2 SECOND-ORDER ENERGY CORRECTION . . . 40
2.2.3 PHASE-SPACE FILLING . . . 41
3.0 REVIEW OF OTHER KINETIC MODELS FOR MICROCAVITY POLARITONS . . . 42
3.1 LOW-DENSITY STUDIES . . . 42
3.2 HIGH-DENSITY STUDIES . . . 45
4.0 EXPERIMENTS . . . 54
4.1 ANGLE-RESOLVED MEASUREMENTS . . . 54
4.1.1 THE EFFECT OF STRESS . . . 63
4.1.2 LINE BROADENING AND LINE NARROWING . . . 65
4.1.3 ERROR ESTIMATES . . . 68
4.2 TIME RESOLVED SPECTROSCOPY . . . 71
5.0 NUMERICAL RESULTS . . . 78
5.1 SIMULATION . . . 78
5.2 MODELING THE EXPERIMENTAL DATA . . . 81
5.2.1 INITIAL FIT WITH CHANGING THE EFFECTIVE SCATTERING CROSSECTION . . . 82
5.2.2 FITS USING POLARITON-ELECTRON SCATTERING AND PIEZOELECTRIC SCATTERING . . . 85
6.0 FUTURE DIRECTIONS AND CONCLUSION . . . 95
6.1 ACCOMPLISHMENTS . . . 95
6.2 WHAT'S NEXT . . . 96
APPENDIX A. KINETICS OF BOSON-BOSON SCATTERING IN A 2D FLAT POTENTIAL . . . 98
A.0.1 Results . . . 99
A.0.2 Conclusion . . . 100
APPENDIX B. FULL CODE . . . 107
B.1 Integrate.c . . . 107
B.2 Constants.h . . . 110

B.3 Parameters.h . . . 111
B.4 Polpara.h . . . 113
B.5 Ecset.h . . . 113
B.6 Getk3.h . . . 114
B.7 dEdk.h . . . 114
B.8 DOS.h . . . 115
B.9 del.h . . . 115
B.10 getf4.h . . . 117
B.11 polcheck.h . . . 118
B.12 polEnergy.h . . . 118
B.13 polfraction.h . . . 118
B.14 pulseflat.h . . . 118
B.15 updatef.h . . . 119
B.16 polpMatx.h . . . 121
B.17 polelMatx.h . . . 123
B.18 polpTAMatx.h . . . 126
B.19 polpTA_PiezoMatx.h . . . 129
B.20 2Dpolscat.h . . . 133
B.21 2Dpolelscat.h . . . 136
B.22 2DpolpLAscat.h . . . 137
B.23 2DpolpTAscat.h . . . 139
B.24 fvsEsave.h . . . 141
B.25 gauleg.h . . . 144
B.26 initiate.h . . . 144
B.27 scat.h . . . 145
APPENDIX C. CODE USER MANUAL . . . 148
C.1 Introduction . . . 148
C.2 Header Functions . . . 149
C.2.1 Inputs . . . 149
C.2.2 Initialization . . . 151

C.2.3 Scattering 152
C.2.4 Updating 153
C.2.5 Renormalization 153
C.2.6 Saving 153
C.2.7 Miscellaneous Functions 154
BIBLIOGRAPHY 156

LIST OF TABLES

1.1 Thicknesses and indicies of refraction for materials used in the rear distributed Bragg reflector. 3

LIST OF FIGURES

1.1 The two shaded regions on each side represent the stacks of material making the distributed Bragg reflectors (DBR's). The thick dark lines in the central part represent groups of quantum wells. The optical intensity of a cavity mode is drawn between the DBR's. The quantum wells are placed at the antinodes. 4

1.2 GaAs quantum well between $Ga_{0.8}Al_{0.2}As$. $k_BT << E_g$ at 4 K implies that the chemical potential is near the middle of the band gap for both materials. The chemical potential is the dashed line halfway between the valence and conduction bands in this illustration. The type of structure shown is known as a Type I heterostructure. 6

1.3 The dashed lines represent the cavity and exciton modes. When brought together there is level repulsion leading to the upper polariton and lower polariton modes, solid lines. For large $k_{\|}$ the modes decouple and become the constituent cavity and exciton modes. 8

1.4 Light gets refracted upon traversing the sample-air interface. $\vec{k}_{\|}$ is conserved. A microcavity has a θ which is only dependent on $\vec{k}_{\|}$ since $k_{z,air}$ is a constant for a microcavity. 12

1.5 The density of states as a function of $k_{\|}$. Large values of k are the density of states for uncoupled excitons. 13

1.6 Momentum and energy are conserved when two particles scatter from k_m. Particles efficiently populate $k = 0$ when k_m is pumped directly. 14

1.7 Trapping polaritons with stress[18]. The gray line across the figure shows the $k = 0$ mode as a function of position. The dip in this line is where the stress is applied and the trap is created. Polaritons created on the side of the trap(blue) can be seen to be migrating toward the bottom of the trap. 16

1.8 A plot showing the critical temperature as a function of the power law of a trapping potential from [20]. 19

1.9 A composite reflectivity measurement showing the high energy edge of the stop band, the upper polariton, and the lower polariton. 20

1.10 The layout for the angle resolved experiments. 21

1.11 Light incident on an AO cell gets diffracted by the sound waves propagating through the cell. 23

1.12 Stressor, sample, mount, and laser. 24

2.1 The angle θ_1 that $\vec{k}_1$ makes to the direction of the difference between $\vec{k}_0$ and $\vec{k}_2$. 28

3.1 From [12], the calculated formation coefficient, C, for the equation $F(E) = Cn_c^2(E)$ where $F(E)$ is formation rate of upper and lower polaritons for a non-resonant pump and $n_c(E)$ is the carrier density. $E = 0$ is the bare exciton energy. 44

3.2 Occupation number vs energy for polaritons. The polariton density for each simulation is given in the upper right hand corner of each graph. The existence of the bottleneck, the peak in the curve, remains when polariton-polariton scattering is considered along with polariton-phonon scattering from [26]. $E = 0$ is the bare exciton energy. The bottleneck is pushed to lower energies with higher density, but never goes to the lowest energy. 51

3.3 Porras, *et al.* [45], showed that numerical simulation suggested the possibility that strongly pumping a material like CdTe, with higher saturation density than GaAs, would result in a large occupation of the lowest energy states. Their pumping density was less than, but on the same order of magnitude, as the saturation density for CdTe. P_x is the pumping rate into the system in $cm^{-2}/100$ ps. P_x is shown for 1, 2, 5, 8, and 15. Notice that the distribution is sloped, not flat, for energies below the bottleneck. 52

3.4 Numerical simulation of polaritons from [48]. GaAs parameters for the effective masses, deformation potential to acoustic phonons, and Coulombic and Pauli exclusion terms. 53
4.1 Optical set up near the cryostat. At the time this picture was taken the cryostat had been replaced by a mirror. The cryostat sits in the background. 55
4.2 A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 1 mW of incident pump power. 57
4.3 A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 6 mW of incident pump power. 58
4.4 A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 24 mW of incident pump power. 59
4.5 A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 35 mW of incident pump power. 60
4.6 A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 80 mW of incident pump power. 61
4.7 CW pumping the side of the stress well. The evolution of the luminescence over five seconds. Thermal effects cause a delay in the build up of the luminescence. The hotter particles also drift farther into the trap since they have a higher average kinetic energy. Each image is integrated over 200 ms. The intensity scale is the same for all images. 62

4.8 A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.05 mW of incident pump power. 63

4.9 A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.2 mW of incident pump power. 64

4.10 A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.4 mW of incident pump power. 65

4.11 A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.6 mW of incident pump power. 66

4.12 A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.8 mW of incident pump power. 67

4.13 Dispersion curve fit to the 1 mW CW laser data shown in Figure 4.2 68

4.14 Occupation of the lower polariton states for different incident pump powers. These were deduced from the spatially integrated images in Figures 4.2-4.6 for CW pumping conditions. 69

4.15 Occupation of the lower polariton states for different incident pump powers. These were deduced from the spatially integrated images in Figures 4.8-4.12 for the quasi-CW pumping condition of 2.4% pump duty cycle. The dotted line represents a T = 90 K Maxwell-Boltzmann distribution and the solid line is for a T = 90 K, $\mu = -0.15$kT Bose-Einstein distribution 70

4.16 The effect of the stress well. From top to bottom the stress is increasing from no stress to resonant stress. Without stress the polaritons are held in high k states. With stress the polaritons are able to make it past the bottle neck. The intensity scales are different for all three plots. 71

4.17 The k = 0 spectrum with quasi-CW pumping of 0.05 mW. 72

4.18 A schematic of the time resolved set up. The beamsplitter splits the pump beam. Part of the pump beam is sent to a delay stage. The rest of the pump beam is incident on the microcavity sample. The solid red line from the microcavity represents luminescence. 75

4.19 The luminescence from the microcavity lasts much longer than the gate pulse. Only a fixed portion of the luminescence mixes inside the BBO with the gate pulse for a given delay. That portion is denoted by the dashed lines. Changing the delay in the gate pulse will sample another point in the microcavity's luminescence. 76

4.20 The integrated intensity from a resolved spectroscopy of the $\vec{k} = 0$ lower polariton with 141 mW of incident pump power above the stop band. 77

5.1 The energy step size as a function of bin number on the mesh. The dispersion curve for the polaritons is plotted on the secondary vertical axis. 79

5.2 The distribution function of the polaritons as it evolved for one set of parameters. Simulated time and the number of iterations are given. 81

5.3 A fit to the CW pumped data using polariton-polariton and polariton-phonon scattering. "A" stands for the coefficient used in front of the polariton-polariton scattering cross-section and "P" is the generation rate used. Simulated plots are shown next to their corresponding experimental pump power. 83

5.4 A fit to the quasi-CW pumped data using polariton-polariton and polariton-phonon scattering. "A" stands for the coefficient used in front of the polariton-polariton scattering cross-section and "P" is the generation rate used. Simulated plots are shown next to their corresponding experimental pump power. 84

5.5 Plot of the coefficient used for the polariton-polariton scattering matrix element as a function of simulated polariton density. 85

5.6 Plot of the simulated generation rates to the corresponding experimental pump powers. The line is a guide for the eye. 87

5.7 Plot of the simulated generation rates to the corresponding experimental pump powers. The line is a guide for the eye. 88

5.8 Plot of the steady state simulated polariton density as a function of simulated generated rate. 89
5.9 Plot of the total simulated free electron density as a function of simulated generated rate. 90
5.10 The final fits to the CW experimental data. In the legend, "T" stands for the simulated lattice temperature, "np" is the simulated polariton density, "ne" is the simulated electron density. Simulated plots are shown next to their corresponding experimental pump power. 91
5.11 The final fits to the quasi-CW experimental data. In the legend, "T" stands for the simulated lattice temperature, "np" is the simulated polariton density, "ne" is the simulated electron density. Simulated plots are shown next to their corresponding experimental pump power. 92
5.12 The steady state results of using (1) polariton-phonon interactions, (2) polaritons interacting with polaritons, electrons and phonons, but no Bose statisitcs, and (3) polaritons interacting with polaritons, electrons, and phonons with Bose statistics included. 94
6.1 The steady state simulated distribution function for the polaritons using the same parameters as the highest generation rate for the quasi-CW data without the free electron-polariton interaction included. 97
A1 Low density energy distribution at scattering times 0, 1, 3, 5 102
A2 The evolution of the fitting parameters, chemical potential and temperature, to the distributions for three different densities. Time is measured in scattering events. The chemical potential is in units of the equilibrium chemical potential and the temperature is in units of the lattice temperature. 103
A3 Scattering rate per particle in the lowest energy bin as as a function of density below the quantum concentration 104
A4 Scattering rate per particle in the lowest energy bin as as a function of density 105
A5 Close up of figure 4. 106
C1 A flow chart of the main code 155

PREFACE

I would like to thank Dr. David Snoke, my advisor, for his patience and support while I prepared this thesis. I would also like to thank the faculty and staff in the Department of Physics and Astronomy without whom I would not have been able to complete this. I also thank those graduate students who I had contact with which made my graduate studies invaluable. Mostly among these is Ryan Balili. I also want to thank three generations of family, from my parents to my sons, for their support through the whole of my academic career.

Vincent E. Hartwell

Pittsburgh, PA

1.0 INTRODUCTION

The material for this thesis was acquired during the quest for a solid state Bose-Einstein Condensation(BEC). While atomic BEC was produced a decade ago, finding a solid state BEC is very relevant. The more systems that can be found to obey Bose-Einstein statistics increases our confidence in the theory. It also provides new interesting phenomena to explore. It is also important to define how composite bosons behave at high densities. On the practical side, a solid state condensate is much easier to create and maintain than an atomic condensate. The temperatures, a few Kelvin as opposed to nanoKelvin, and trapping methods, a simple pin as opposed to large magnetic and optical traps, are much less heroic. Also, practical applications such as ones that would use the interference of two condensates, will need to be at room temperature. While the experiments for this thesis were done at four degrees Kelvin, it is reasonable to believe that similar systems can be engineered to allow BEC at room temperature. A publication regarding room temperature polariton lasers in GaN microcavities has already appeared[1].

This thesis is a study of the dynamics of two dimensional quasi-particles, called polaritons, which exist in a semiconductor microcavity structure. This chapter gives an overview of what microcavity polaritons are and what experimental equipment was used to study them. Chapter 2 defines the theory behind the numerical simulations used to simulate the experiments, while a review of numerical studies is provided in Chapter 3. The experimental measurements (Chapter 4) and the numerical simulations (Chapter 5) regarding the steady state of these quasi-particles will be provided. A nonthermalized steady state results from the interplay between the short lifetime of the polaritons and the rate at which excitons are able to scatter into the polaritonic region of the dispersion curve. The result being that at most densities the lower energy polariton states have lower occupation, a negative temperature.

Chapter 6 gives final conclusions and direction for the continued work on polaritons.

Conceptually, the microcavity structures used in our experiments are composed of two main parts. One part consists of two Bragg reflectors which makes up a cavity for photon modes, the other part consists of quantum wells which confine the exciton modes. Strong coupling of the photon modes to the exciton modes allows polariton modes to exist.

1.1 CAVITY MODES

An optical cavity is created when two reflective surfaces are placed such that their surface normals are parallel. The general term for optical cavities is an etalon, of which the Fabry-Perot interferometer and the laser cavity are examples. The spacing between the end mirrors in a cavity causes a resonant condition for specific wavelengths. Standing waves are created for those optical frequencies given by $\nu_m = cm/2nL$, where c is the speed of light, n is the effective index of refraction for the space between the mirrors, L is the distance between the mirrors, and m is an index that begins at 1, the fundamental. It is simple to show that $\Delta\nu = \nu_{m+1} - \nu_m = c/2nL$. This equation can be used to calculate the mode spacing for the cavity.

A semiconductor microcavity is made by epitaxial growth of a distributed Bragg reflector (DBR) on a substrate. For our microcavities the substrate is GaAs. The rear Bragg reflector is made of twenty groups of alternating regions of semiconductor materials, in this case, AlAs, GaAs, and $Ga_{0.8}Al_{0.2}As$. Table 1.1 provides the thickness and indicies of refraction for each material. Indices are quoted for photon energies of 1.62 eV. The quarter wave stacks are thus made of AlAs and $Ga_{0.8}Al_{0.2}As$ and a transition material, GaAs, is used between them to make the structure grow with fewer defects. The internal part of the optical cavity which contains the quantum wells is grown next and will be discussed in the next section. The top part of the cavity is grown last. For our sample, the top Bragg reflector consists of sixteen groups of alternating layers of $Ga_{0.8}Al_{0.2}As$ (579 $\AA$ thick) and AlAs, (672 $\AA$ thick) and capped with a final layer of 579 $\AA$ thick $Ga_{0.8}Al_{0.2}As$.

Thus, through multiple-beam interference of quarter-wave stacks, two mirrors are cre-

material	thickness(Å) of layer	index of refraction
AlAs	672	3.0 [2][3]
GaAs	30	3.6 [4]
$Ga_{0.8}Al_{0.2}As$	548	3.5 [5]

Table 1.1: Thicknesses and indicies of refraction for materials used in the rear distributed Bragg reflector.

ated. The cavity has a region of high reflectivity called the stop band. This region spans energies from around 1.55 eV to around 1.72 eV for the sample studied here. Within this stop band, only light that is resonant with the cavity can have an appreciable field inside the cavity. For photon energies around 1.6 eV, our Bragg reflectors are spaced $L = \frac{3}{2}\lambda$ apart. Such a spacing creates a cavity resonance which is called a cavity mode, (see Figure 1.1 for an illustration). The actual value for the resonant energy varies somewhat at different positions on the sample. The growth procedure does not create a sample of uniform thickness but a sample that is slightly wedged. As the thickness changes over the sample, the resonant mode's wavelength shifts.

However, there is only one resonant mode within the stop band at any particular position on the sample. This is because $\Delta\nu = c/2nL = c/3n\lambda = \nu/3$. Since $\Delta E = h\Delta\nu$, then the energy spacing is $E_{cavity}/3$. For all values of energy in the stop band the next resonant energy is well outside the stop band.

The energy of a photon is given by

$$E = \frac{\hbar c k}{n}, \tag{1.1}$$

where k is the wavenumber. The cavity wavenumber, k_z, is constrained in the growth direction since the only allowed wavevector in the z-direction is $k_z = 2\pi/\lambda = 3\pi/L$ and L is a constant. Then the energy can be written as

$$E_{cavity} = \frac{\hbar c}{n}\sqrt{k_z^2 + k_\parallel^2} = \frac{\hbar c}{n}\sqrt{\left(\frac{3\pi}{L}\right)^2 + k_\parallel^2} \tag{1.2}$$

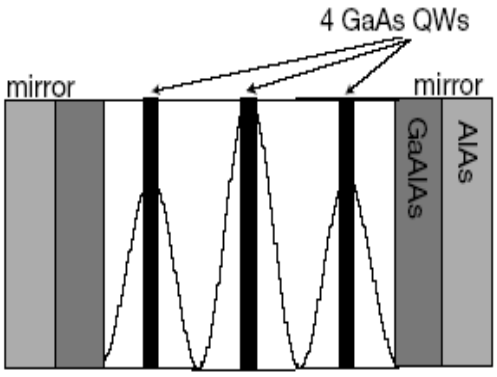

Figure 1.1: The two shaded regions on each side represent the stacks of material making the distributed Bragg reflectors (DBR's). The thick dark lines in the central part represent groups of quantum wells. The optical intensity of a cavity mode is drawn between the DBR's. The quantum wells are placed at the antinodes.

with $k_{\|}$ the in-plane wavenumber of the photon. The energy of the cavity mode at any particular spot on the sample is only a function of $k_{\|}$.

1.2 QUANTUM WELL EXCITONS

The ground state of a semiconductor has its valence band full of electrons and its conduction band empty. A photon of energy larger than the band gap can promote an electron from the valence band to the conduction band. The semiconductor is then in an excited state, having an electron in the conduction band and a hole in the valence band. Such a state is analogous to an atom that has been ionized. As with an atom, there are also discrete states of lower energy in a semiconductor. These are bound states known as excitons.

The exciton can be shown[6] to have a dispersion relationship relative to the band gap energy equal to

$$E = -\frac{E^*_{Ry}}{n^2} + \frac{\hbar^2|\vec{k}|^2}{2M}, \tag{1.3}$$

where E^*_{Ry} is the binding energy, n is the principal quantum number, and the last term is the kinetic energy of the exciton with M the sum of the effective masses of the electron and hole that make up the exciton. The binding energy is similar to that of the hydrogen atom, but is adjusted for the different effective masses of the electron and hole. Additionally, the charges of the electron and hole are screened. Taking into account the dielectric constant of the material, ϵ, gives

$$E^*_{Ry} = \frac{13.6\text{eV}}{m_0}\frac{\mu}{\epsilon^2}, \tag{1.4}$$

with m_0 the mass of the bare electron and μ the reduced effective mass of the exciton. E^*_{Ry} can be used to calculate the size of the exciton, as the hydrogen atom's size can be calculated from its energy. The exciton Bohr radius, a^{ex}_B, is

$$a^{ex}_B = \frac{e^2}{4\pi E^*_{Ry}\epsilon\epsilon_0}. \tag{1.5}$$

So far this discussion has been about excitons in bulk material. The excitons studied in this thesis exist in quantum wells between our cavity's distributed Bragg reflectors. As shown in Figure 1.1, there are three groups of quantum wells. Each group consists of alternating layers of 70 $\mathring{A}$ thick GaAs followed by 30 $\mathring{A}$ thick $Ga_{0.8}Al_{0.2}As$ barriers to make four quantum wells. $Ga_{0.8}Al_{0.2}As$ has a band gap, ~ 1.8 eV, and GaAs has a band gap, ~ 1.5 eV, when the temperature is near 4 Kelvin[7]. In a quantum well, the GaAs in this case, the excitons are confined due to this difference in band gap. This shifts the minimum energy of the exciton if the confining thickness is small enough. The $|\vec{k}|^2$ in Equation 1.3 gets replaced with

$$|\vec{k}|^2 = \left(\frac{n\pi}{L}\right)^2 + k^2_{\parallel}, \tag{1.6}$$

where n is the principal number of the quantum level in the well, and L is the width of the quantum well. Like the cavity mode, the quantum well exciton's energy is only a function

of $k_{\|}$. The lowest energy of the kinetic part of the exciton's energy in Equation 1.3 then becomes,

$$E^{QW} = \frac{\hbar^2 \pi^2}{2ML^2}. \tag{1.7}$$

Our excitons are confined to the 7.0 nm thick quantum wells, for which the first allowed state is shifted to 1.6 eV, which has a corresponding free-space optical wavelength near 770 nm.

A schematic for a quantum well is shown in Figure 1.2. This configuration is known as a Type I heterostructure where both the valence band and the conduction band are confined in the same material.

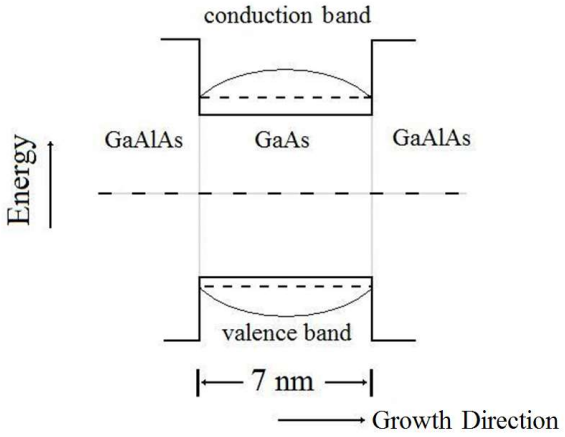

Figure 1.2: GaAs quantum well between $Ga_{0.8}Al_{0.2}As$. $k_B T << E_g$ at 4 K implies that the chemical potential is near the middle of the band gap for both materials. The chemical potential is the dashed line halfway between the valence and conduction bands in this illustration. The type of structure shown is known as a Type I heterostructure.

Exciton binding energies are increased when the exciton is confined to a quantum well since the bands become nonparabolic, there is Coulombic coupling of excitons in different subbands, there is valence-band mixing of the different hole states, and also the effect the

different dielectric constants of the well and barrier materials have[8]. The binding energy as a funtion of well thickness can be modeled

$$E_{Ry}^{QW} = E_{Ry}^{*} \frac{L_0}{L} \tag{1.8}$$

where L_0 is a parameter that characterizes the rate at which the energy changes with quantum well width, L. The binding energy of excitons in GaAs is 10 meV in 7 nm quantum wells[8], up from 4 meV in bulk. For a 10 meV binding energy this implies that the excitonic radius is about 100 Å. This is important since the exciton size is larger than the quantum well width, which emphasizes that the exciton must be contained in the plane of the quantum well and should therefore behave as if it were in a two dimensional system. This is also important because the planar density of particles, n, must satisfy

$$n a_B^2 << 1, \tag{1.9}$$

for the particles to be treated as a weakly interacting Bose gas. The exciton is a composite boson since together the electron and hole give the exciton an integer value for spin. For the excitons not to see the fermionic structure of their nearest neighbors and behave as bosons, the above condition on the density must be met. For our case, this limiting density is $n \sim 2.0 \times 10^{10}$ cm^{-2} per quantum well for $n a_B^2 < 0.1$. Our samples have twelve quantum wells so that total densities, $n \sim 2.4 \times 10^{11}$ cm^{-2} are acceptable.

1.3 POLARITONS

Our system of interest consists of an optical cavity in which the photon mode is nearly resonant with the exciton mode that exists in quantum wells within the cavity. When the cavity photon mode couples to the excitonic mode, polaritons are created. If we plot the dispersion curves for the photons and the excitons when there is no coupling, we would see that the modes overlap at zero detuning. Detuning is the difference in energy between the two modes at $k_{\|} = 0$. However, when the modes couple, there is level repulsion of the modes. The resulting plots, which I will give a derivation for, are shown in Figure 1.3. The dashed

lines represent the uncoupled modes and the solid lines represent the lower polariton and upper polariton. The lower polaritons are the quasiparticles with which we are concerned. Note that there can also be phonon polaritons when phonon modes couple to photon modes, but we will not consider these.

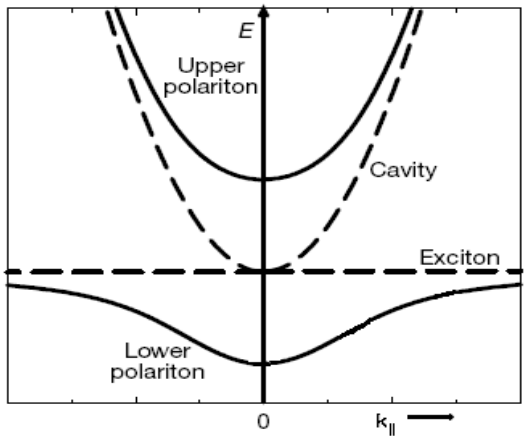

Figure 1.3: The dashed lines represent the cavity and exciton modes. When brought together there is level repulsion leading to the upper polariton and lower polariton modes, solid lines. For large $k_{\|}$ the modes decouple and become the constituent cavity and exciton modes.

Since the two modes are coupled, the energy in the system sloshes back and forth between the two. This coupling results in a splitting of the energy levels, or in the language of quantum mechanics, repulsion of the levels. Two different methods of deriving the polariton dispersion curve will be described. Both give the same answer. For the first method, we start with the photon dispersion relation,

$$n = \frac{ck}{\omega}. \tag{1.10}$$

The index of refraction, n, is the square root of the dielectric function $\epsilon(\omega)$. Thus,

$$\epsilon(\omega) = \frac{c^2k^2}{\omega^2}, \tag{1.11}$$

where k is for that for a cavity, $\sqrt{\left(\frac{3\pi}{L}\right)^2 + k_{\|}^2}$ with L the cavity length. Now we introduce a system that has optical absorption, the excitons. The dielectric function of the system is then described by the equation

$$\epsilon(\omega) = \epsilon_\infty + \frac{q^2 N}{mV} \frac{1}{\omega_0^2 - \omega^2}, \tag{1.12}$$

where ω_0 is the exciton resonance frequency and q^2N/mV is known as the oscillator strength of the system, q is the charge, m is the mass, and N/V is the density of available oscillators. A damping term in the denominator of the right hand side has been neglected. Setting Equation 1.11 and Equation 1.12 equal and rearranging, we get the quartic equation in terms of ω,

$$\omega^4 - \left(\omega_0^2 + \frac{q^2 N}{mV\epsilon(\infty)} + \frac{c^2k^2}{\epsilon(\infty)}\right)\omega^2 + \omega_0^2 \frac{c^2k^2}{\epsilon(\infty)} = 0. \tag{1.13}$$

We take the two positive solutions of this equation since energies are positive. This leads to

$$\omega^2 = \frac{1}{2}\left(\omega_0^2 + \frac{q^2 N}{mV\epsilon(\infty)} + \frac{c^2k^2}{\epsilon(\infty)} \pm \sqrt{\left(\omega_0^2 - \frac{q^2 N}{mV\epsilon(\infty)} - \frac{c^2k^2}{\epsilon(\infty)}\right)^2 - 4\omega_0^2 \frac{c^2k^2}{\epsilon(\infty)}}\right). \tag{1.14}$$

When the exciton energy is nearly resonant with the cavity modes we can take the approximation $\omega_0 \approx ck/n$ which leads to

$$E_{u,l} = \frac{1}{2}\left(\hbar\omega_0 + \frac{\hbar ck}{\epsilon(\infty)} \pm \sqrt{\left(\frac{\hbar ck}{\epsilon(\infty)} - \hbar\omega_0\right)^2 + \left(\frac{\hbar^2 q^2 N}{4mV\epsilon(\infty)}\right)}\right). \tag{1.15}$$

The $E_{u,l}$ are commonly referred to as the upper(lower) polariton. The constant energy, $\hbar\Omega_R = \hbar\sqrt{q^2N/(4mV\epsilon(\infty))}$ is a measure of the splitting of the energy levels at resonance. It is usually referred to as the Rabi splitting of the system. The Rabi frequency, Ω_R, is the rate at which the energy is moving back and forth between the exicitons and photons.

Additional information is obtained about the polariton eigenstates by solving the problem with quantum mechanics. We have the exciton, a dipole, interacting with the photons, an

external electric field. Collectively, dipoles provide a polarization. The interaction between the polarization and electric field gives the interaction Hamiltonian[6],

$$H_{int} = -\int d^3r \vec{P} \cdot \vec{E}. \tag{1.16}$$

The quantized electric field is writen as

$$\vec{E}(\vec{r}) = -i \sum_{\vec{k}'} \sqrt{\frac{\hbar\omega_{\vec{k}'}}{2\epsilon_\infty V}} \left(a_{k'} e^{i\vec{k}'\cdot\vec{r}} - a_{k'}^\dagger e^{-ik'\cdot r} \right), \tag{1.17}$$

with $a_k^\dagger(a_k)$ are photon creation(destruction) operators. The polarization is given by

$$\vec{P}(\vec{r}) = \frac{qN}{V} \sqrt{\frac{\hbar}{2mN\omega_0}} \sum_k \left(b_k e^{ik\cdot r} + b_k^\dagger e^{-ik\cdot r} \right), \tag{1.18}$$

with $b_k^\dagger(b_k)$ are exciton creation(destruction) operators. Making these substitutions and performing the integral for H_{int}, with $\omega_k \approx \omega_0$, the full Hamiltonian for the cavity modes (a_k), the exciton modes (b_k), and the interaction is given by

$$H = \sum E_c(\vec{k}) a_{\vec{k}}^\dagger a_{\vec{k}} + \sum E_x(\vec{k}) b_{\vec{k}}^\dagger b_{\vec{k}} + i \sum \hbar\Omega_R \left(b_{\vec{k}}^\dagger a_{\vec{k}} - a_{\vec{k}}^\dagger b_{\vec{k}} \right). \tag{1.19}$$

This Hamiltonian is non-diagonal because of the coupling term. The mixed terms can be removed through a unitary transformation

$$\begin{pmatrix} l_{\vec{k}} \\ u_{\vec{k}} \\ l_{\vec{k}}^\dagger \\ u_{\vec{k}}^\dagger \end{pmatrix} = \begin{pmatrix} X_{\vec{k}} & C_{\vec{k}} & 0 & 0 \\ -C_{\vec{k}} & X_{\vec{k}} & 0 & 0 \\ 0 & 0 & X_{\vec{k}} & C_{\vec{k}} \\ 0 & 0 & -C_{\vec{k}} & X_{\vec{k}} \end{pmatrix} \begin{pmatrix} b_{\vec{k}} \\ a_{\vec{k}} \\ b_{\vec{k}}^\dagger \\ a_{\vec{k}}^\dagger \end{pmatrix} \tag{1.20}$$

where $C_{\vec{k}}$ and $X_{\vec{k}}$ are Hopfield coefficients[9]. Working through this transformation it is found that these coefficients are given by

$$X_{\vec{k}}^2 = \frac{1}{2} \left(\frac{E_{ck} - E_{xk}}{\sqrt{(E_{xk} - E_{ck})^2 + (2\hbar\Omega_R)^2}} + 1 \right) \tag{1.21}$$

and

$$C_{\vec{k}}^2 = 1 - X_{\vec{k}}^2. \tag{1.22}$$

$X_{\vec{k}}^2$ is the fraction of the polariton that is excitonic, and $C_{\vec{k}}^2$ is the fraction that is photonic. At zero detuning, the lower polariton is half exciton and half photon and the polariton becomes increasingly excitonic with larger $k_\parallel$. The eigenvalues found by this method are the same as those found in the classical derivation, Equation 1.15.

In the graph of the dispersion relationship (Figure 1.3), the cavity mode is curved instead of a straight near $k_\parallel = 0$, as a free photon would be. This occurs because $|\vec{k_c}| = \sqrt{k_z^2 + k_\parallel^2}$ and k_z is a constant. The cavity mode then has an effective mass given by the general relationship for particles,

$$m^* = \frac{\hbar^2}{2\frac{\partial E}{\partial k^2}}. \tag{1.23}$$

The exciton also has a mass, but it is much larger than the photon's mass, thus it appears flat on the scale provided in that figure.

The constant k_z provides us an easy way to study the microcavity polaritons. In the bulk case, k_z is not a constant for the polaritons. Even though $\vec{k}_\parallel$ is a conserved optical quantity across the boundary for two and three dimensional systems, the light leaving the surface of a three dimensional system at a given angle can contain a continuum of $k_\parallel$ and k_z wavevectors. In the two-dimensional case, the single value for k_z allows for direct measurement of $\vec{k}_\parallel$ by detecting the angle of the light emitted from the surface of the sample, as illustrated in Figure 1.4. By measuring the intensity and energy of the photoluminescence at various angles from the sample, the polaritons can be studied in k-space.

The polaritons, being the actual eigenstates of the system, have a mass comparable to the cavity photon mass, which is on the order of 10^{-4} smaller than the exciton mass. This has some implications. The density of states is given by a general formula,

$$D(E) = \frac{Ag}{(2\pi)^2} \int_0^\infty 2\pi k dk \delta(E_k - E), \tag{1.24}$$

where A is the sample area and g is a degeneracy factor. Taking the derivative of the relationship between energy and wavenumber, $E = \hbar^2 k^2 / 2m$ gives,

$$dE = \frac{\hbar^2}{2m} k dk. \tag{1.25}$$

Substituting this into Equation 1.24 and using $\int dE\delta(E) = 1$ gives,

$$D(E) = \frac{Ag}{2\pi}\frac{m}{\hbar^2}. \tag{1.26}$$

This equation shows that the density of states is proportional to mass. The density of states is a constant in two dimensions when the mass is constant, but for polaritons near $k = 0$, it is orders of magnitude smaller than the bare exciton as shown in Figure 1.5. This plot was calculated by using Equation 1.15 and Equation 1.23 in 1.26. As pointed out in Figure 1.3, at large $k_{\|}$ the constituent modes decouple and become cavity and exciton modes again.

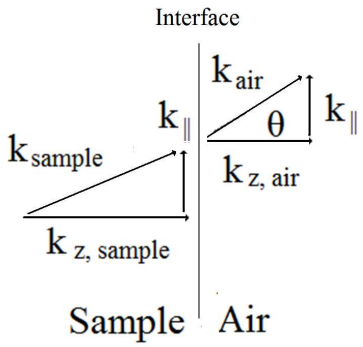

Figure 1.4: Light gets refracted upon traversing the sample-air interface. $\vec{k}_{\|}$ is conserved. A microcavity has a θ which is only dependent on $\vec{k}_{\|}$ since $k_{z,air}$ is a constant for a microcavity.

The light mass is one of the main reasons for studying polaritons. A phase transition, which would be a Kosterlitz-Thouless phase transition[10] in a potential-free geometry, is expected when the particle density is around the quantum concentration,

$$n_Q = \frac{gmkT}{\pi\hbar^2}. \tag{1.27}$$

The required particle density is linear in mass and we expect that the concentration of polaritons can be much smaller than excitons for a phase transition at the same temperature. Even at room temperature, two orders of magnitude higher temperature than when excitons

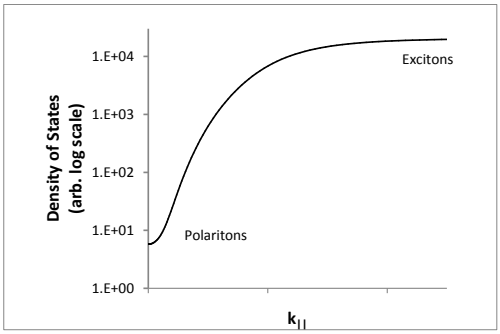

Figure 1.5: The density of states as a function of $k_{\|}$. Large values of k are the density of states for uncoupled excitons.

have the possiblity of condensing, the required concentration for a polariton phase transition is still one hundred times less than that for bare excitons. This is good because it allows the required concentration to remain low enough that the particles behave as bosons (Equation 1.9).

There are some possible difficulties when using polaritons to study BEC. The main one is time. The lifetime of the polaritons is only a few picoseconds. There may not be enough time for long range order to build up between the particles and allow for a true BEC[11]. This was the earliest concern. It was predicted and experimentally observed that there was a bottleneck effect in the scattering toward the lowest polariton states[12][13]. The lower polaritons simply escaped the cavity before they could accumulate at the bottom of the dispersion curve. Not until higher polariton densities were attained was it realized that the bottleneck could be overcome. Higher polariton densities were attained by increasing the number of quantum wells in the samples. This allowed for more intense pumping without saturating the exciton density. Our sample had twelve quantum wells, reducing the effective exciton density per quantum well for a given pump power by a factor of twelve.

A way to bypass the bottleneck is to use "magic angle" scattering[14]. The "magic angle"

scattering occurs because there is a point on the dispersion curve, k_m, that bisects a line drawn from the $k_{\|} = 0$ to the region where the lower polariton starts to become excitonic, see Figure 1.6. Two particles at this point can then scatter, one to $k_{\|} = 0$ and the other to $k_{\|} = 2k_m$, conserving energy and momentum. Experimentally, this requires the laser to be incident on the sample at the "magic angle" with the corresponding wavelength. However, this method leads one to question if any coherence in the $k_{\|} = 0$ state is due to the pump coherence tuned to k_m.

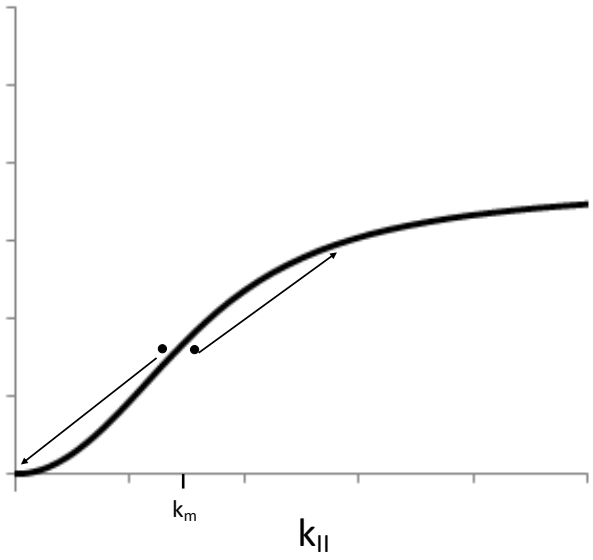

Figure 1.6: Momentum and energy are conserved when two particles scatter from k_m. Particles efficiently populate $k = 0$ when k_m is pumped directly.

Another problem with polaritons is that intense pumping can take the system from a strong coupling regime to a weak one. Photonic lasing then occurs[15]. The density at which this occurs is related to the Bohr radius of the excitons, $n \sim 1/a_B^2$.

There is another phenomenon related to the excitonic Bohr radius and particle density. Experimentally, a blueshift in the emission can occur. A blueshift occurs because of the repulsive energy of the excitonic part of the polaritons[16]. This may cause a problem in

a trap. It may blueshift the bottom of the trap enough that the polaritons spread out, decreasing the particle density at the bottom of the trap. Smaller excitons, as in some other materials, should reduce this effect.

Despite all these issues, microcavity polaritons behave like particles. They have mass and can move around. Questions about their dynamics immediately arise. Will they condense under suitable conditions? Can they, how do they, and how long does it take for them to come to thermal equilibrium from a nonequilibrium state? If they cannot thermalize can they come to some steady state? Can they exhibit any spontaneous coherence in a steady state? This thesis studies the polariton dynamics experimentally and numerically to try to answer a few of these questions.

1.4 POLARITON TRAPPING

In the experiments for this thesis the polaritons are trapped. While the numerical models shown later do not consider trapping, I mention it here as part of the experimental introduction and its importance with regard to creating a two dimensional BEC. Negoita, *et al.* showed that when quantum wells were stressed, a spatial trap for excitons could be created[17], and Balili, *et al.*[18] showed that polaritons could be trapped by this method as well, Figure 1.7. The trapping occurs because the stress geometry stretches the lattice of the sample on the side opposite of where the stress is applied. The band gap is related to the lattice spacing. When the lattice spacing increases, the band gap decreases, thereby shifting the energy of the exciton as well.

In these experiments the experimental site on the sample is chosen such that the exciton mode is positively detuned from the cavity mode. Positive detuning means the exciton mode is at a higher energy than the cavity mode. The stress applied to the site then depresses the exciton energy until it is resonant with the cavity mode. This resonant point has the highest energy splitting for the polaritons. Nearby positions, which are not as resonant, do not experience the same magnitude of splitting and their lower polaritons are at higher energies. Thus a spatial trap is created at the point of maximum stress.

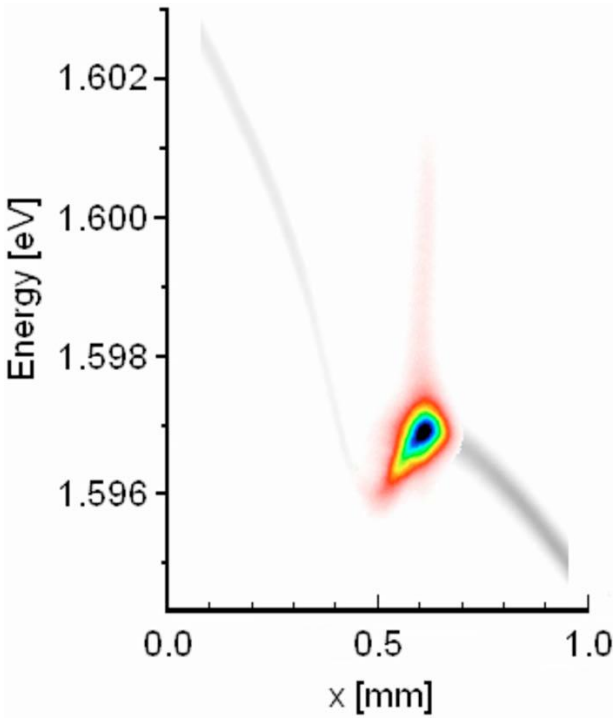

Figure 1.7: Trapping polaritons with stress[18]. The gray line across the figure shows the $k = 0$ mode as a function of position. The dip in this line is where the stress is applied and the trap is created. Polaritons created on the side of the trap(blue) can be seen to be migrating toward the bottom of the trap.

The number of particles in a low density system is given by

$$N = \int dE\ D(E)\ f(E) \tag{1.28}$$

where $D(E)$ is the density of states and $f(E)$ is the occupation for the type of statistics under consideration, Bose statistics in our case. If the total number of this integral does not

equal the total number of particles in the system then the additional particles must be in the ground state, measured by N_0, giving,

$$N = N_0 + \int dE\ D(E)\ f(E). \tag{1.29}$$

This integral diverges whenever the temperature is greater than zero and $D(E_{min}) \neq 0$. Figure 1.5 shows that the density of states for the polaritons is not zero for $E_{min}(k_{\|} = 0)$. Hohenberg has shown that no long range order can exist in such a system, if the particles are free[19]. It has also been shown[20, 21] that two-dimensional systems in a power-law potential can achieve a condensate. This is because the density of states in the integral in Equation 1.29 allows the integral to converge.

Each degree of freedom for a harmonic oscillator has energy

$$E_i = \left(n_i + \frac{1}{2}\right)\hbar\omega_i. \tag{1.30}$$

In a two-dimensional system with a harmonic trap, the energy levels are given by

$$E = (n_x + n_y + 1)\hbar\omega = (n+1)\hbar\omega, \tag{1.31}$$

with n_x and n_y the integer quantum numbers (0, 1, 2,...) for the independent motions. The value of n is also an integer since n_x and n_y are integers. For any value of n there are $n+1$ ways to choose n_x and n_y and the degeneracy of each energy level is simply given by $n+1$. The number of particles in a system is then given by the sum of the degeneracy of each state times the occupation of each state,

$$N = \sum_n \frac{n+1}{e^{((n+1)\hbar\omega-\mu)/k_BT} - 1}. \tag{1.32}$$

When the average energy, $E = k_BT$, is much larger than the energy spacing, $\hbar\omega$, we take the Stringari-Pitaevskii limit[22], $N \to \infty, \omega \to 0$ such that $N\omega^2 =$ constant. We multiply the sum by $\hbar\omega^2$[23],

$$N(\hbar\omega)^2 = \sum_n \hbar\omega \frac{(n+1)\hbar\omega}{e^{((n+1)\hbar\omega-\mu)/k_BT} - 1}. \tag{1.33}$$

Since $\Delta E = \hbar\omega$, we can change the sum over n to a sum over E_n and in the limits discussed above, switch to the integral

$$N = \frac{1}{(\hbar\omega)^2} \int dE \frac{E}{e^{\frac{E-\mu}{k_B T}} - 1}. \tag{1.34}$$

Through a change of variables, $\epsilon = E/k_B T$ and letting $\mu = 0$, the integral becomes

$$N = \left(\frac{k_B T}{\hbar\omega}\right)^2 \int_0^\infty d\epsilon \frac{\epsilon}{e^\epsilon - 1} \tag{1.35}$$

Performing the integral provides the result for the critical number,

$$N_c = 1.645 \left(\frac{k_B T}{\hbar\omega}\right)^2 \tag{1.36}$$

An alternative derivation was given by Bagnato and Kleppner [20]. They showed that the highest critical temperature in a two-dimensional system was for a potential with an r^2 dependence, a harmonic potential, Figure 1.8. Our method of trapping creates a harmonic potential for the polaritons.

1.5 EXPERIMENTAL SETUPS

A general setup is described here to introduce the main equipment used. Any deviations will be described in more detail in the chapter on experiments.

The sample, as has been described in Sections 1.1 and 1.2, grown by L. Pfeiffer and K. West at Bell Labs, was cut and the substrate chemically thinned by polishing with bromine and methanol. Etching helps to facilitate the stress trapping method discussed above by reducing the sample's thickness from 0.5 microns to 0.15 microns. It was then placed in a Janis cryostat and cooled to near four Kelvin by continuously flowing helium gas over the sample. We used an 18 W Coherent Verdi to pump a Coherent MIRA Ti:Sapphire laser. The MIRA was run in CW mode and tuned to the first main gap in the sample's stop band (see Figure 1.9). For some of the experiments, output of the MIRA was focused through an AO Modulator, Brimrose TEM-200-50-633, that was driven at 1 kHz by driver model FFA-200-B1(50). The electrical pulse to the AO cell had a duty cycle of 2.4%. Thus, the diffracted

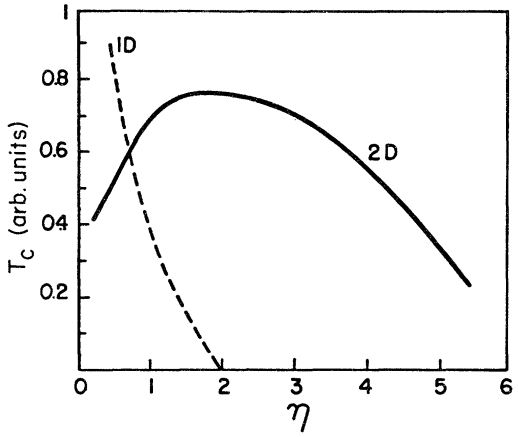

Figure 1.8: A plot showing the critical temperature as a function of the power law of a trapping potential from [20].

laser light from the AO cell made a choppped beam. The pump beam was then transmitted through a continuously variable attenuator, Edmund Optics NT41-960. The attenuator allows for controlling the power incident on the sample. The beam was then focused on to a spot size of 30 μm. The focus of the beam was then incident on the sample at an oblique angle of incidence of 15 degrees. Photoluminescence from the sample was collected with a fiber optic bundle, Edmund Optics NT39-368, mounted on an arm that could rotate about an axis centered on the sample. The entrance to the fiber optic was positioned 21 cm from the sample. With an aperture of 1.6 mm, the fiber optic collected light from the sample in a 7 mrad solid angle. During an angle-resolved experiment the arm was rotated in 1 degree ($\sim$ 17 mrad) steps. The other end of the fiber optic was rigidly mounted and aligned with a Spectra Pro spatial imaging spectrometer. A CCD camera, Cascade512B, captured the optical information at the output of the spectrometer. The CCD images were saved to a

computer. Figure 1.10 shows a general set up. This figures leaves out optics such as lenses, mirrors, and filters. While these are important optics, they do not add anything significant to the diagram or to understanding the experimental procedure.

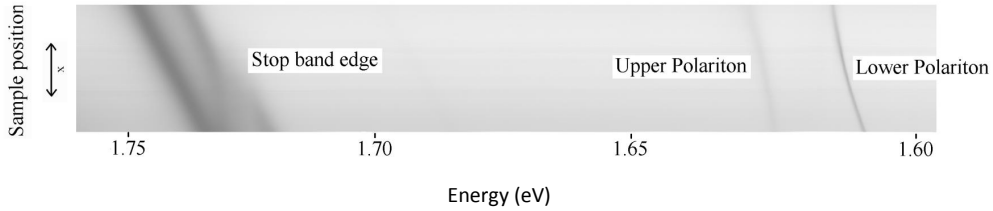

Figure 1.9: A composite reflectivity measurement showing the high energy edge of the stop band, the upper polariton, and the lower polariton.

1.5.1 MIRA

The Coherent MIRA is a diode-pumped Ti:Sapphire laser capable of producing 200 fs pulses at 76 MHz repetition with 1.5 W output and $\Delta\nu$ ~20 nm bandwidth. The self-mode-locking mode of the MIRA has a narrow transverse spot relative to the non mode-locked mode. By

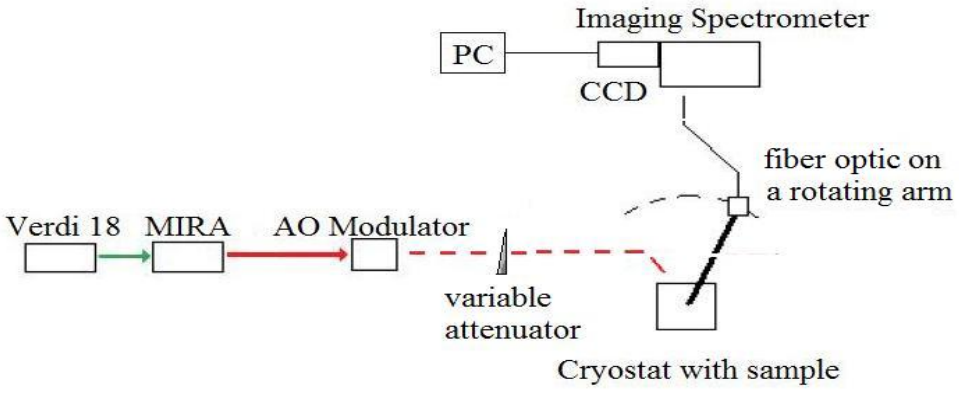

Figure 1.10: The layout for the angle resolved experiments.

allowing the internal beam path to conduct the non-mode-locked mode, the pulsed mode is suppressed and CW output is achieved. This CW mode has a full width half maximum bandwidth of 0.1 nm.

Depending on the optic set in a Ti:Sapphire, it is capable of lasing from 700 nm to 1000 nm. The output wavelength is determined by a birefringent filter at Brewster's Angle which only passes a relatively small fraction of these wavelengths without losses. The laser is tunable by correct orientation of this filter. For the data presented here, the MIRA was set to lase around 718 nm. The exact wavelength may vary. It depends on where the stop

band ends for the sample position being excited by the laser.

1.5.2 AO MODULATION

We use an acoustic-optic modulator to chop our pulses for some of the experiments. Our modulator is made of a photo-elastic TeO_2 crystal that is interfaced to a piezoelectric quartz crystal. An electric pulse is applied to the quartz causing sound waves to emanate from it which get transmitted to the TeO_2. Sound waves are variations in the density of the crystal. To first order, the change in index of refraction is proportional to this change in density. The sound waves therefore create a variation in index of refraction that acts as a grating, as illustrated in Figure 1.11. The spacing of the grating is determined by the wavelength of the sound. Our acousto-optic modulator has a frequency, $\nu = 200$ MHz, and the sound waves travel at a velocity, $v = 4200$ m/s. Light incident on this grating will diffract according to the grating equation $\lambda = 2a\sin(\theta)$ where $a = \lambda_{sound}/2 = v/2\nu$ is the spacing of the grating. Since the sound wave is travelling it also imparts some momentum to the lightwave and thus changes the lightwave's frequency. However, the change in wavelength of our lightwave is negligible for the type of experiment done here.

1.5.3 CRYOSTAT AND STRESSOR INSERT

We used a Janis optical cryostat. This device is capable of cooling down to the λ-point of liquid He. A sample is loaded into the cryostat by an insert fabricated by Ryan Balili. The top of the insert has a micrometer which is attached to a spring loaded tube. The spring loaded tube is a piston inside another tube. This outer tube is rigidly mounted to the micrometer base. The spring loaded tube has a needle on the end where the sample holder is. The sample holder is attached to the outer tube by springs. The sample holder clamps around the outer edge of the sample and the needle is positioned just above the sample. When the micrometer is turned, the needle pushes against the sample. The springs in the sample mount help to reduce the rate at which stress is applied, making it less likely to break a sample with overstress. Figure 1.12 depicts the sample mount geometry. An additional micrometer was built into the insert to allow vertical translation of the whole insert without

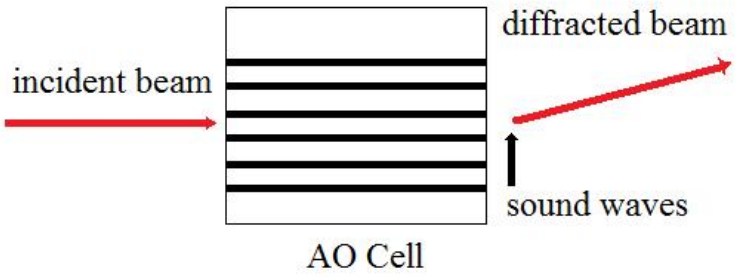

Figure 1.11: Light incident on an AO cell gets diffracted by the sound waves propagating through the cell.

moving the cryostat or changing the stress. This is important because as the system cools down, the insert undergoes thermal contraction. Thus, if optical alignment of the system was done before cooling, the alignment could be easily recovered with this second micrometer.

Additionally, the insert has an electrical connection to a silicon diode. Monitoring the temperature near the sample is accomplished with this device. Its resistance changes by a known amount with temperature. By flowing a constant current through it, the voltage across it could be measured and compared to a known voltage-versus-temperature chart.

The bottom of the cryostat has four windows. We use one view window to inject our exciting laser and use the same window to transmit the photoluminescence from the micro-cavity.

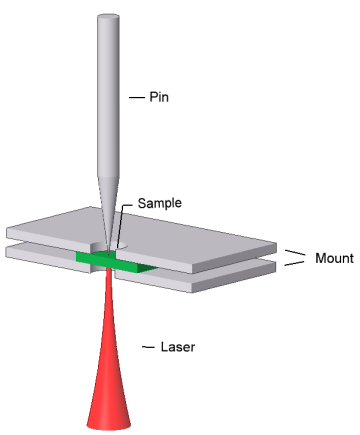

Figure 1.12: Stressor, sample, mount, and laser.

1.5.4 IMAGING SPECTROMETER AND CCD CAMERA

Angle-resolved data was taken with a fiber optic mounted on an arm that rotated about an axis below the cryostat. The arm could be positioned so that its rotation was centered on the sample. Photoluminescence from the sample could be collected by the fiber optic and fed into a spectrometer. We used a Specta Pro Imaging spectrometer. The optical properties of the spectrometer preserve the vertical spatial intensity upon transmission through the device. This particular property of the spectrometer is not relevant when using the fiber optic since spatial information is lost in going through the fiber optic. A grating in the spectrometer, with 1800 lines/mm, disperses in a horizontal direction the light that makes it through the front slit. Thus the output of the spectrometer contains information about the intensities of different energies in the horizontal direction while showing how the energies are spatially related in the vertical direction. Figure 1.7 and Figure 1.9 are examples of spectrograms that were recorded by the CCD Camera placed at the output of the spectrometer. For Figure 1.7

the axes have been switched with vertical measuring energy as opposed to in the horizontal direction for Figure 1.9. In both cases, light from the sample has been imaged directly on to the front slit of the spectrometer instead of going through the fiber optic.

Our Photometrics Cascade 512B CCD camera has 16 μm pixels in a 512 x 512 array. The CCD can be exposed for as little as 1 ms and requires 30 ms to read out the data. It has a split screen so that one image can be moved into a readout area while another image is being exposed, which for long enough exposure times allows for continuous exposure. Typically we did not push the limits of the readouts as we exposed it for 300 ms or more.

2.0 THE MODEL

2.1 SCATTERING

The theoretical model which I present accounts for three types of scattering: polariton-polariton scattering, polariton-phonon scattering, and polariton-electron scattering.
A derivation of the scattering equations used is provided in the following subsections.

2.1.1 POLARITON-POLARITON INTERACTIONS

The scattering rates for the simulation follow Fermi's Golden Rule,

$$\frac{\partial f_{\vec{k}_0}}{\partial t} = \frac{2\pi}{\hbar} \sum_{\vec{k}_f} |\langle \vec{k}_f | V_{int} | \vec{k}_i \rangle|^2 \delta(E_{final} - E_{initial}), \tag{2.1}$$

where two particles scatter from state $|\vec{k}_i\rangle = |\vec{k}_0 \vec{k}_3\rangle$ to state $\langle \vec{k}_f| = \langle \vec{k}_2 \vec{k}_1|$. This is done through a scattering interaction V_{int}, which for a two-body elastic scattering process is

$$V_{int} = |M| b^\dagger_{\vec{k}_1} b^\dagger_{\vec{k}_2} b_{\vec{k}_3} b_{\vec{k}_0}, \tag{2.2}$$

where M is the matrix element for the interaction and the $b^\dagger$'s(b's) are creation(destruction) operators. $|V_{int}|$ becomes, after the b's operate on a bosonic Fock state,

$$|V_{int}| = |M| \sqrt{n_{\vec{k}_0} n_{\vec{k}_3} (n_{\vec{k}_2} + 1)(n_{\vec{k}_1} + 1)} \tag{2.3}$$

where $n_{\vec{k}_m}$ is the occupation number of the state $\vec{k}_m$. Squaring V_{int} and putting it back into Fermi's Golden Rule gives the scattering out of state $\vec{k}_0$ for bosons as

$$\frac{\partial n_{\vec{k}_0}}{\partial t} = \frac{2\pi}{\hbar} \sum_{\vec{k}_1 \vec{k}_2} |M(|\vec{k}_0 - \vec{k}_2|)|^2 n_{\vec{k}_0} n_{\vec{k}_3} [1 + n_{\vec{k}_1}][1 + n_{\vec{k}_2}] \delta(E(\vec{k}_0) + E(\vec{k}_3) - E(\vec{k}_2) - E(\vec{k}_1)), \tag{2.4}$$

where momentum has been conserved through $\vec{k}_3 + \vec{k}_0 = \vec{k}_1 + \vec{k}_2$. The sums over $\vec{k}_1$ and $\vec{k}_2$ can be converted to integrals by taking the thermodynamic limit:

$$\frac{\partial n_{\vec{k}_0}}{\partial t} = \frac{S^2}{(2\pi)^3\hbar}\int d^2\vec{k}_1 d^2\vec{k}_2 |M(|\vec{k}_0 - \vec{k}_2|)|^2 \times$$

$$n_{\vec{k}_0} n_{\vec{k}_3}[1 + n_{\vec{k}_1}][1 + n_{\vec{k}_2}]\delta(E(\vec{k}_0) + E(\vec{k}_3) - E(\vec{k}_2) - E(\vec{k}_1)). \tag{2.5}$$

where S is the area of the sample. We assume the system is isotropic in k-space. Then $n_{\vec{k}} = f(E(\vec{k}))$, which is independent of θ, and $n(E) = n_{|k|} = f(E(\vec{k}))D(E(\vec{k}))dE$, where $D(E)dE$ is the number of states with dE of E. The change to the number of particles within dE of E_0 per unit time for scattering out of a state is

$$\frac{\partial n(E_0)}{\partial t} = \frac{S^2 D(E_0)}{(2\pi)^3\hbar} dE_0 \int d^2\vec{k}_1 d^2\vec{k}_2 |M(|\vec{k}_0 - \vec{k}_2|)|^2 \times$$

$$f(E_0) f(E_3)[1 + f(E_1)][1 + f(E_2)]\delta(E_0 + E_3 - E_2 - E_1). \tag{2.6}$$

This equation can be analytically simplified to reduce the numerical work necessary to solve it. We need to integrate over $\vec{k}_1$ and $\vec{k}_2$, or over the variables, $k_1 = |\vec{k}_1|$, θ_1, $k_2 = |\vec{k}_2|$, and θ_2. The delta function arguments, E_1, E_2, and E_3 depend on these variables. We pick one variable, θ_1, to integrate out the delta function. We could pick any of the variables, but only $E_3(|\vec{k}_3|^2)$ is dependent on the angles. By integrating over an angle, the derivation is simpler. We can integrate θ_1 relative to any direction and we choose that direction to be $\vec{k}_2 - \vec{k}_0$, as shown in Figure 2.1.

Since the delta function is not explicitly defined in terms of θ_1 we must make a change of variables, or use the following identity,

$$\delta(g(\theta)) = \sum \frac{\delta(\theta - a)}{|g'(a)|}. \tag{2.7}$$

where the sum is over all values of a such that the argument of the delta function is zero and the derivative with respect to θ_1 of the argument of the delta function is not zero.

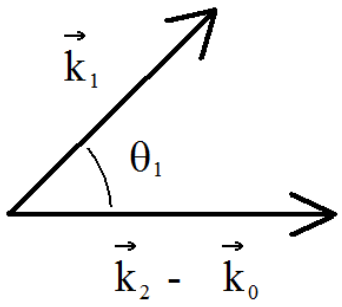

Figure 2.1: The angle θ_1 that $\vec{k}_1$ makes to the direction of the difference between $\vec{k}_0$ and $\vec{k}_2$.

Then, with $E(|\vec{k}_3|^2)$ written out explicitly in terms of θ_1,

$$g(\theta_1) = E_0 + E_3(|\vec{k}_2 - \vec{k}_0|^2 + k_1^2 + 2|k_1||\vec{k}_2 - \vec{k}_0|\cos(\theta_1)) - E_1 - E_2 \tag{2.8}$$

and using the chain rule for differentiation,

$$|g'(\theta_1)| = \frac{\partial E_3}{\partial(k_3^2)} 2|k_1||\vec{k}_2 - \vec{k}_0|\sin(\theta_1). \tag{2.9}$$

For polaritons, the expression for $\partial E/\partial(k^2)$ is reasonably compact and easier to derive than $\partial E/\partial k$. For this reason, the expression, $\partial E/\partial(k^2)$, is used several places in this thesis instead of $\partial E/\partial k$. Equation 1.23 is usually written with $\partial^2 E/\partial k^2$, for example. $\partial E/\partial(k^2)$ is given by,

$$\frac{\partial E}{\partial(k^2)} = \frac{\hbar^2\left[\frac{E_{xk}}{m}(E_{ck}^2 - E_k^2) + \frac{c^2}{n^2}(E_{xk}^2 - E_k^2)\right]}{2E_k(E_{xk}^2 + E_{ck}^2 + (2\hbar\Omega)^2 - 2E_k^2)}, \tag{2.10}$$

where E_k refers to the polariton and E_{ck} and E_{xk} refer to the uncoupled cavity and uncoupled exciton modes respectively, m is the excitonic mass, n is the effective index of refraction in the microcavity, and Ω is the Rabi frequency of the coupling.

Now, we set $g(\theta_1) = 0$ and derive the values of a by solving for θ_1. This results in,

$$\delta(\theta_1 - a) \Rightarrow a = \cos^{-1}\left[\frac{k^2(E_3) - |\vec{k}_2 - \vec{k}_0|^2 - |k_1|^2}{2|k_1||\vec{k}_2 - \vec{k}_0|}\right], \tag{2.11}$$

where $k^2(E_3)$ is the wavenumber squared for energy $E_3 = E_1 + E_2 - E_0$. $k(E_3)$ is written as k_3 from now on.

The argument of the Arccos must be less than 1 and greater than -1. Each of these limits is taken separately after replacing $|\vec{k}_2 - \vec{k}_0|^2$ with $|\vec{k}_2|^2 + |\vec{k}_0|^2 - 2|\vec{k}_2||\vec{k}_0|\cos\theta_2$, where θ_2 is measured relative to θ_0. We solve the resulting two equations for the limits of integration over θ_2 and obtain,

$$\theta_{2min} = \cos^{-1}\left[\frac{k_0^2 - k_1^2 + k_1k_3}{k_0k_2}\right] \tag{2.12}$$

$$\theta_{2max} = \cos^{-1}\left[\frac{k_0^2 - k_1^2 - k_1k_3}{k_0k_2}\right]. \tag{2.13}$$

Using the relationship $\sin(\cos^{-1}(m)) = [1-m^2]^{\frac{1}{2}}$,

$$|g'(\theta_1)| = \frac{\partial E_3}{\partial(k_3^2)} 2|k_1||\vec{k}_2 - \vec{k}_0| \left(1 - \left[\frac{k^2(E_3) - |\vec{k}_2 - \vec{k}_0|^2 - |k_1|^2}{2|k_1||\vec{k}_2 - \vec{k}_0|}\right]^2\right)^{\frac{1}{2}}. \tag{2.14}$$

Substituting this result and replacing $d^2\vec{k}$ with $kdkd\theta$ in Equation 2.6 and integrating over θ_1 gives,

$$\frac{\partial n(E_0)}{\partial t} = \frac{S^2 D(E_0)}{(2\pi)^3\hbar} dE_0 \int k_1 dk_1 k_2 dk_2 \theta_2 \frac{|M(|\vec{k} - \vec{k}_2|)|^2 f(E_0) f(E_3)[1+f(E_1)][1+f(E_2)]}{\frac{\partial E_3}{\partial(k_3^2)} 2|k_1||\vec{k}_2 - \vec{k}|\left(1 - \left[\frac{k^2(E_3) - |\vec{k}_2 - \vec{k}|^2 - |k_1|^2}{2|k_1||\vec{k}_2 - \vec{k}|}\right]^2\right)^{\frac{1}{2}}}. \tag{2.15}$$

This is converted to an integral over energy using,

$$dk = \frac{dE}{\frac{\partial E}{\partial(k^2)} 2k}. \tag{2.16}$$

We obtain

$$\frac{\partial n(E_0)}{\partial t} = \frac{S^2 D(E_0)}{4(2\pi)^3\hbar} dE_0 \int dE_1 dE_2 d\theta_2 \times$$

$$\frac{|M(|\vec{k} - \vec{k}_2|)|^2 f(E_0) f(E_3)[1+f(E_1)][1+f(E_2)]}{\frac{\partial E_3}{\partial(k_3^2)} \frac{\partial E_1}{\partial(k_1^2)} \frac{\partial E_2}{\partial(k_2^2)} 2|\vec{k}_1||\vec{k}_2 - \vec{k}_0|\left(1 - \left[\frac{k^2(E_3) - |\vec{k}_2 - \vec{k}_0|^2 - |k_1|^2}{2|k_1||\vec{k}_2 - \vec{k}_0|}\right]^2\right)^{\frac{1}{2}}}. \tag{2.17}$$

The denominator can be simplified to give,

$$\frac{\partial n(E_0)}{\partial t} = \frac{S^2 D(E_0)}{8(2\pi)^3\hbar} dE \int dE_1 dE_2 d\theta_2 \times$$

$$\frac{|M(|\vec{k}-\vec{k}_2|)|^2 f(E_0) f(E_3)[1+f(E_1)][1+f(E_2)]}{\frac{\partial E_3}{\partial(k_3^2)}\frac{\partial E_1}{\partial(k_1^2)}\frac{\partial E_2}{\partial(k_2^2)} 2|\vec{k_0}||\vec{k}_2|\Big([cos(\theta_{2min})-cos(\theta_2)][(cos(\theta_2)-cos(\theta_{2max})]\Big)^{\frac{1}{2}}}, \quad (2.18)$$

where θ_{2min} and θ_{2max} are given above in Equation 2.12 and Equation 2.13.

The integral for $\partial n(E)/\partial t$ is now evaluated numerically. The integral over θ_2 can be transformed so as to be done with Gaussian Quadrature. Gaussian Quadrature allows for better results with fewer points and thus speeds up the calculation. Additionally, the transformation avoids the poles that result from the limits of integrating over θ_2. In the simulation, the code for calculating the abscissa points and weights for the Gaussian Quadrature are taken from Numerical Recipes in C [24]. This book also gives a good review of the theory behind Gaussian Quadrature.

The polariton-polariton interaction matrix element, $M(|\vec{k_0}-\vec{k}_2|)$, was studied by Ciuti, *et al.*[25] In this interaction, only the excitonic components of the polaritons are interacting. A factor, $X_{k,k'}$, is used in the interaction model to account for this. $X_{k,k'}$ stands for the product of the X_k Hopfield coefficients for the participating polaritons (see Equation 1.21). There are four polaritons involved here, so $X_{k,k'} = X_{\vec{k}_0} X_{\vec{k}_1} X_{\vec{k}_2} X_{\vec{k}_3}$. M is given by

$$M = \frac{1}{S}\sum_{\vec{k}\vec{k'}} X_{k,k'} V_{\vec{k}-\vec{k'}} \phi_{\vec{k}} \phi_{\vec{k'}} (\phi_{\vec{k}}^2 - \phi_{\vec{k}}\phi_{\vec{k'}}). \quad (2.19)$$

The interaction potential ,$V_{\vec{k}-\vec{k'}}$, is proportional to $1/|\vec{k}-\vec{k'}|$ for two dimensional Coulomb interactions, and $\phi_{\vec{k}} = \sqrt{8\pi a_B^2/S}(1+k^2a_B^2)^{-3/2}$. An approximation for M was determined by Tassone and Yamamoto [26] to be

$$M \sim 6 X_{k,k'} E_B \frac{a_B^2}{S}. \quad (2.20)$$

This value, representing hard core scattering, is used for the matrix element for polariton-polariton scattering in the models for this thesis.

Scattering *into* a given k-state can be calculated with an equation similar to Equation 2.18. The difference is in the statistical part. We replace the statistical factors as follows:

$$f(E_0)f(E_3)[1+f(E_1)][1+f(E_2)] \rightarrow f(E_1)f(E_2)[1+f(E_0)][1+f(E_3)]. \quad (2.21)$$

2.1.2 POLARITON-LONGITUDINAL ACOUSTICAL PHONON INTERACTIONS

The derivation given in the last section was for a four body process in two dimensions. The derivation in this section is for a three-body process in which the polaritons are constrained to two dimensions, but the phonons are three-dimensional. The polariton-phonon model is based on the exciton-phonon interaction [27], which itself is based on the electron-deformation potential interaction.[6] The deformation potential interaction has the form

$$M(|\vec{k},\vec{q}|) = iX_{k,k'}\sqrt{\frac{\hbar\left(q_{\parallel}^2+q_z^2\right)^{\frac{1}{2}}}{2\rho V u}}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right] \tag{2.22}$$

where ρ, V, u, and q are the material density, volume, longitudinal sound velocity, and phonon wavenumber, respectively. The deformation potentials are given for each band as a_e and a_h. These values are taken to be the bulk value for GaAs, $a_e = -7$ eV and $a_h = 2.7$ eV [27]. The $X_{k,k'}$ is only the product of two Hoppfield Coefficients since there are only two polaritons involved. The other factors are overlap integrals between the excitons in the quantum wells and the phonons in the sample using the envelope function approximation[8, 12, 27, 28, 29]:

$$I_{e(h)}^{\perp}(q_z) = \frac{8\pi^2}{L_z q_z\left(4\pi^2 - L_z^2 q_z^2\right)}\sin\left(\frac{L_z q_z}{2}\right) \tag{2.23}$$

and

$$I_{e(h)}^{\parallel} = \left[1+\left(\frac{m_{h(e)}}{2M}\left|q_{\parallel}\right|a_B\right)^2\right]^{-\frac{3}{2}}, \tag{2.24}$$

where L_z is the quantum well thickness.

As with the polariton-polariton interactions, we use Fermi's Golden rule for scattering. One process is scattering out of state $\vec{k}$ while creating a phonon:

$$\frac{\partial n_{\vec{k}}}{\partial t} = \frac{2\pi}{\hbar}\sum_{\vec{k}_1,\vec{q}_z}|M(|\vec{k},\vec{q}|)|^2 n_{\vec{k}}[1+n_{\vec{k}_1}][1+n_{\vec{q}}]\delta(E(\vec{k})-E(\vec{q})-E(\vec{k}_1)). \tag{2.25}$$

The in-plane momentum has been conserved through $\vec{q}_{\parallel} = \vec{k} - \vec{k}_1$ with $\vec{q} = \sqrt{\vec{q}_z^2 + \vec{q}_{|parallel}}$.

Then changing the sums to an integral gives

$$\frac{\partial n_{\vec{k}}}{\partial t} = \frac{1}{\rho u 8\pi^2}\int d^2\vec{k}_1 dq_z X_{k,k_1}\left(q_\parallel^2+q_z^2\right)^{\frac{1}{2}}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z)-a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right]^2\times$$

$$n_{\vec{k}}[1+n_{\vec{k}_1}][1+n_{\vec{q}}]\delta(E(\vec{k})-E(\vec{q})-E(\vec{k}_1)). \tag{2.26}$$

We replace $d^2\vec{k}_1$ with $k_1 dk_1 d\theta_1$. Also, we assume the system is isotropic in k-space as we did for the polariton-polariton scattering. Therefore, the change to the number of particles within dE of energy E per unit time is

$$\frac{\partial n(E)}{\partial t} = \frac{D(E)}{\rho u 8\pi^2}dE\int k_1 dk_1 d\theta_1 dq_z X_{k,k_1}\left(q_\parallel^2+q_z^2\right)^{\frac{1}{2}}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z)-a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right]^2\times$$

$$f(E)[1+f(E_1)][1+F(\hbar u|\vec{q}|)]\delta(E-\hbar u|\vec{q}|-E_1). \tag{2.27}$$

Here, $F(q)$ is the Bose distribution for phonons.

Instead of integrating out the delta function by integrating over an angle we integrate over q_z. We again make use of Equation 2.7 and use the form:

$$\delta(g(q_z)) = \sum\frac{\delta(q_z-a)}{|g'(a)|}, \tag{2.28}$$

so that

$$g(q_z) = E-E_1-\hbar u\sqrt{(|q_\parallel|^2+q_z^2)} \tag{2.29}$$

and

$$|g'(q_z)| = \frac{\hbar u q_z}{\sqrt{(|q_\parallel|^2+q_z^2)}}. \tag{2.30}$$

We set $g(q_z)=0$ and derive the values of a by solving for q_z. The result is

$$a = \left[\frac{(E-E_1)^2}{(\hbar u)^2}-q_\parallel^2\right]^{\frac{1}{2}}. \tag{2.31}$$

We now obtain

$$|g'(a)| = \frac{(\hbar u)^2\left[\frac{(E-E_1)^2}{(\hbar u)^2}-|\vec{k}-\vec{k}_1|^2\right]^{\frac{1}{2}}}{E-E_1}. \tag{2.32}$$

This derivative cannot equal zero and must be real, so we write,

$$\frac{(E-E_1)^2}{(\hbar u)^2} - |\vec{k}-\vec{k}_1|^2 > 0. \tag{2.33}$$

Physically, this means that the created phonon has some momentum in the z-direction. The elastic wave propagates in both the positive and negative z directions. Since, $|\vec{k}-\vec{k}_1|^2 = |\vec{k}|^2 + |\vec{k}_1|^2 - 2|\vec{k}||\vec{k}_1|\cos\theta_1$, we make this substitution and solve for $\cos\theta_1$,

$$\cos\theta_1 > \left(\frac{k^2 + k_1^2 - \frac{(E-E_1)^2}{\hbar^2 u^2}}{2kk_1}\right) < 1. \tag{2.34}$$

In the denominator of $g'(a), E \neq E_1$, since if the two energies were the same there would not be a phonon created.

Using these results in Equation 2.27 and integrating over q_z gives

$$\begin{aligned}\frac{\partial n(E)}{\partial t} &= \frac{D(E)}{\rho u 8\pi^2} dE \int k_1 dk_1 d\theta_1 \frac{X_{k,k_1}(E-E_1)^2}{(\hbar u)^3\left[\frac{(E-E_1)^2}{(\hbar u)^2} - |\vec{k}-\vec{k}_1|^2\right]^{\frac{1}{2}}} \times \\ &\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(a) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(a)\right]^2 f(E)[1+f(E_1)][1+F(E-E_1)]. \end{aligned}\tag{2.35}$$

Now, converting to integrate over energy, we get

$$\begin{aligned}\frac{\partial n(E)}{\partial t} &= \frac{D(E)}{\rho u 16\pi^2} dE \int \frac{dE_1 d\theta_1}{\frac{\partial E_1}{\partial(k_1^2)}} \frac{X_{k,k_1}(E-E_1)^2}{(\hbar u)^3\left[\frac{(E-E_1)^2}{(\hbar u)^2} - |\vec{k}-\vec{k}_1|^2\right]^{\frac{1}{2}}} \times \\ &\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(a) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(a)\right]^2 f(E)[1+f(E_1)][1+F(E-E_1)]. \end{aligned}\tag{2.36}$$

This equation is used for scattering out of a state $\vec{k}$ while emitting a phonon. There are three other possibilities for phonon interaction. These are scattering out of a state $\vec{k}$ while absorbing a phonon, scattering into a state $\vec{k}$ while emitting a phonon, and scattering into a state $\vec{k}$ while absorbing a phonon. Two of these processes are the inverse of the other two and that symmetry is used as a check on the numerical calculation.

2.1.3 POLARITON-TRANSVERSE ACOUSTICAL PHONON INTERACTIONS

The polaritons interact with transverse acoustical phonons as well. Essentially the scattering is the same as for scattering with longitudinal phonons except transverse phonon values are used. Snoke *et al.*,[30] showed that the effective deformation potential, Ξ, for transverse phonons can be given by

$$\Xi = \left(\frac{4}{5} * \left(b^2 + \frac{d^2}{2}\right)\right)^{\frac{1}{2}}, \tag{2.37}$$

with b and d as deformation potentials in the Pikus-Bir notation[31]. Measurements on GaAs show the holes to have $b = 1.8$ eV and $d = 5.4eV$[32]. The electrons have $b = 0$ and $d = 0$ since the conduction band is nondegenerate.

There are two directions of polarization for transverse phonons. A factor of two is included in the polariton-transverse phonon scattering calculation.

2.1.4 POLARITON-PHONON INTERACTION BY PIEZOELECTRICTY

The squeezing of the lattice due to a phonon can cause a local electric field. This effect is called piezoelectricity. The matrix element for polaritons interacting with phonons through piezoelectricity is given by[6],

$$M(|\vec{k},\vec{q}|) = X_{k,k'}\frac{e}{4\pi\epsilon}\sum_{\lambda}\sum_{ijl} e_{ijl}\frac{q_i q_j}{q^2}\eta_{(q\lambda)l}\sqrt{\frac{\hbar}{2\rho V\omega_{q\lambda}}}\left[I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z) - I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right] \tag{2.38}$$

where ρ, V, and q are the material density, volume, and phonon momentum, respectively, η is a unit vector with polarization in the l'th direction, e_{ijl} is the piezoelectric tensor for the material being considered, and the $I's$ have been defined in Equation 2.23 and Equation 2.24.

When two or more terms describe interactions of the same particle types, e.g. acoustic phonons with polaritons in this section and the previous sections, the terms must be added before squaring and using them in Fermi's Golden Rule. However, the deformation potential is an imaginary term while the piozoelectric term is real. Thus, they are completely out of phase with each other and there are no mixed terms upon squaring.

Squaring the above M gives, using u, the speed of sound, and $\omega = \hbar u q$,

$$M^2 = \frac{X_{k,k'}^2}{32\pi^2 \rho V u} \sum\sum_{ijl} e_{ijl}^2 \frac{\eta_{(q\lambda)l}^2 e^2}{\epsilon^2} \frac{q_i^2 q_j^2}{q^5} \left[I_e^{\parallel}(|(\vec{q}|) I_e^{\perp}(q_z) - I_h^{\parallel}(|\vec{q}|) I_h^{\perp}(q_z) \right]^2 \tag{2.39}$$

For GaAs, e_{ijl} only has three non-zero values, $e_{14} = e_{25} = e_{36} = -0.16$ C/m^2 [33]. Reduced notation has been used here[6]. Since all three indices must be different, then the polaritons only have piezoelectric interaction with transverse phonons. Performing the sum over $i, j,$ and l gives,

$$M^2 \propto \frac{2}{q^5} \left(q_x^2 q_y^2 + q_x^2 q_z^2 + q_y^2 q_z^2 \right). \tag{2.40}$$

In cylindrical coordinates this becomes,

$$M^2 \propto \frac{2}{q^5} \left(q_z^2 q_{\parallel}^2 + q_{\parallel}^4 \cos^2\theta_q \sin^2\theta_q \right). \tag{2.41}$$

The angle, θ_q, is measured with respect to the polariton wavevector $\vec{k}$ in the $q_{\parallel}$ plane. To simplify the numerical calculation I find the average magnitude that the trigonometric terms could have, which is

$$\left\langle \cos^2\theta_q \sin^2\theta_q \right\rangle = \frac{1}{2\pi} \int d\theta_q \cos^2\theta_q \sin^2\theta_q = \frac{1}{8}. \tag{2.42}$$

The total matrix element squared for piezoelectric scattering is then

$$M_{piezo}^2 = \frac{X_{k,k'}^2}{16\pi^2 \rho V u} \frac{e_{14}^2 e^2}{\epsilon^2} \frac{q_{\parallel}^2 \left(q_z^2 + \frac{q_{\parallel}^2}{8} \right)}{q^5} \left[I_e^{\parallel}(|(\vec{q}|) I_e^{\perp}(q_z) - I_h^{\parallel}(|\vec{q}|) I_h^{\perp}(q_z) \right]^2. \tag{2.43}$$

2.1.5 POLARITON-OPTICAL PHONON INTERACTIONS

The numerical work for this thesis does not use a Frohlich interaction; the polaritons are at low temperature. Optical phonons in GaAs have $\hbar\omega \cong 36$ meV, which is much greater than k_BT. The code is capable of modeling such an interaction, though. The analytical manipulation of the scattering equation is very similar to the previous sections. For scattering out of state $\vec{k}$ for bosons while creating an optical phonon we use:

$$\frac{\partial n_{\vec{k}}}{\partial t} = \frac{2\pi}{\hbar} \sum_{\vec{k}_1,\vec{q}_z} |M(|\vec{k},\vec{q})|^2 n_{\vec{k}}[1+n_{\vec{k}_1}][1+n_{\vec{q}}]\delta(E(\vec{k})-E(\vec{q})-E(\vec{k}_1)), \tag{2.44}$$

where the matrix element is given by

$$M(|\vec{k},\vec{q}|) = iX_{k,k'}\sqrt{\frac{2\pi e^2\hbar\omega_{LO}}{(\vec{q}_\parallel^2+q_z^2)V}}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)^{\frac{1}{2}}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right] \tag{2.45}$$

and

$$I_{e(h)}^{\perp}(q) = \frac{8\pi^2}{L_z q\,(4\pi^2 - L_z^2 q^2)}\sin\left(\frac{L_z q}{2}\right) \tag{2.46}$$

and

$$I_{e(h)}^{\parallel} = \left[1+\left(\frac{m_{h(e)}}{2M}\left|q_\parallel\right|a_B\right)^2\right]^{-\frac{3}{2}}. \tag{2.47}$$

An optical phonon has an almost flat dispersion relationship so we take the optical phonon energy as a constant, $E(\vec{q}) = \hbar\omega_{LO}$.

Changing the sums to integrals gives

$$\frac{\partial n_{\vec{k}}}{\partial t} = \frac{e^2\omega_{LO}}{2\pi}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)\int d^2\vec{k}_1 dq_z \frac{X_{k,k'}}{(q_z^2+q_\parallel^2)}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(q_z) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(q_z)\right]^2 \times$$

$$n_{\vec{k}}[1+n_{\vec{k}_1}][1+n_{\vec{q}}]\delta(E(\vec{k})-E(\vec{q})-E(\vec{k}_1)), \tag{2.48}$$

where the in-plane momentum has been conserved through $\vec{q}_\parallel = \vec{k}-\vec{k}_1$.

The integration with the delta function is the same as for acoustic phonons, Equations (2.28) - Equations (2.34), and the final result is

$$\frac{\partial n_{\vec{k}}}{\partial t} = \frac{e^2 u}{2\pi\omega_{LO}}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)\int \frac{dE_1 d\theta_1}{\frac{\partial E_1}{\partial(k_1^2)}} X_{k,k'}\left[a_e I_e^{\parallel}(|(\vec{q}|)I_e^{\perp}(a) - a_h I_h^{\parallel}(|\vec{q}|)I_h^{\perp}(a)\right]^2 \times$$

$$n_{\vec{k}}[1+n_{\vec{k}_1}][1+n_{\vec{q}}]. \tag{2.49}$$

Here, $E - E_1$ has been replaced with $\hbar\omega_{LO}$ and

$$a = \sqrt{\left(\frac{\omega_{LO}}{u}\right)^2 - q_\parallel^2}, \tag{2.50}$$

with $\omega_{LO} = 1.07$ x 10^{13} Hz for GaAs.

2.1.6 FREE ELECTRON-POLARITON INTERACTIONS

The matrix elements for direct and exchange interactions for free electron-polariton scattering have been calculated in the literature [34]. This paper shows that the direct term for electron-exciton scattering is much smaller than the exchange term for $\Delta k < 1/a_B$, and $m_e = m_h$. This is reasonable since it can be expected for the free electron to interact equally with the electron and the hole of the exciton. The exchange term provided by [34] was difficult to implement. We set out to derive our own form of the equation.

We start with the 2D excitonic wavefunction, using the symbol k to mean $k_\parallel$,

$$\phi(k) = \sqrt{\frac{2a_B^2}{\pi}}\frac{1}{\left(1+(ka_B)^2\right)^{3/2}}. \tag{2.51}$$

The Hamiltonian term for electron-electron exchange is given by

$$H_{ex} = \frac{1}{2A}\frac{e^2}{\epsilon\left(|\Delta k|+\kappa\right)} c^\dagger_{k-\Delta k} c^\dagger_{k+\Delta k} c_k c_k. \tag{2.52}$$

Here, κ is a screening parameter[35] and the $c_k^\dagger(c_k)$ are creation(destruction) operators for electrons.

The Debye-Hŭckel screening parameter is given by,

$$\kappa = -\frac{\mathrm{e}^2}{2\epsilon_\infty}\sum_{\vec{k}}\frac{\partial f(E(\vec{k}))}{\partial E}. \tag{2.53}$$

Substituting a Fermi-Dirac thermalized distribution for $f(E)$ and performing the sum by converting it to an integral results in the two-dimensional Debye screening formula,

$$\kappa = \frac{\mathrm{e}^2}{2\epsilon_\infty}\frac{n}{k_B T}, \tag{2.54}$$

where n is the two-dimensional density of electrons, k_B is Boltzmann's constant and T is the electron temperature.

Using the exciton wavefunction for the probability amplitude of each electron state, the matrix element, $M = \langle f|H_{ex}|i\rangle$, becomes the integral,

$$\begin{aligned} M &= \frac{e^2}{\epsilon}\frac{a_B^2}{(2\pi)^2}\int d^2k_e \frac{1}{\left(\left(k_2^2 + k_e^2 + \Delta k_2^2 - 2k_2\cdot k_e + 2k_2\cdot\Delta k_2 - 2k_e\cdot\Delta k_2\right)^{1/2} + \kappa\right)} \times \\ &\quad \frac{1}{\left(1 + \left(k_e^2 + \Delta k_2^2 + 2k_2\cdot\Delta k_2\right)^2 a_B^2\right)^{3/2}}\frac{1}{(1 + k_e^2 a_B^2)^{3/2}}. \end{aligned} \tag{2.55}$$

We simplify this by assuming the dot products average to zero. Also, we assume the integration replaces k_e with $1/a_B$ and provides a multiplier of $2/a_B^2$ as determined by numerical integration. The result is

$$M = \frac{\sqrt{2}Ae^2}{4\pi^3\epsilon}\frac{1}{((k_2^2 + (1/a_B^2) + \Delta k_2^2)^{1/2} + \kappa)}\frac{1}{(2 + \Delta k^2 a_B^2)^{3/2}}. \tag{2.56}$$

This element is placed into Fermi's Golden Rule as was done in the previous sections. The distribution function for the particles that the polaritons interact with is replaced with the distribution function for electrons. This is because the electrons are fermions and not bosons. If high density is considered, the statistical part in Fermi's Golden Rule is changed, such that the factor $(1 + n)$ becomes $(1 - n)$.

2.2 ENERGY CORRECTIONS

The dispersion curve shown in Figure 1.3 is the dispersion that a single exciton-polariton in the system would have. Once another particle is introduced into the system, there is an interaction between the particles. If the density of particles is high enough, there can be a noticable change in the dispersion curve.

The simulation can take into account the energy shifts discussed in the following three subsections and calculate a new dispersion relationship. The scattering rates depend on the density of states of the particles, which in turn, depend on the dispersion relationship. Without these shifts all but the statistical parts of the integrals can be done once. When these energy shifts are taken into account, all parts of the scattering rate must be recalculated for each time step and the whole calculation proceeds much slower. The results shown in Chapter 5 do not use these energy corrections.

2.2.1 FIRST-ORDER ENERGY CORRECTION

Two polaritons will interact through their excitonic components. To first order the Hamiltonian can be written as

$$H = \sum_k \hbar\omega_k b_k^\dagger b_k + \frac{1}{2S}\sum_{kk'q} V_{int}(q) b_{k+q}^\dagger b_{k'-q}^\dagger b_k b_{k'}, \tag{2.57}$$

where S is the sample area.

In taking the expectation value of this equation, the only terms that survive are when the momentum is conserved. In the following we separate out the term for $q = 0$. The expectation value of Equation (2.57) is given by

$$\langle H\rangle = \langle|\sum_k \hbar\omega_k b_k^\dagger b_k - \frac{1}{2S}\sum_k V_{int}(0) b_k^\dagger b_k + \frac{1}{2S}\sum_{kk'} V_{int}(|k-k'|) b_k^\dagger b_k b_{k'}^\dagger b_{k'}|\rangle \tag{2.58}$$

This equation can be written with the k's and k''s separated and rearranged, giving

$$\langle H\rangle = \langle|\sum_k \left[\hbar\omega_k - \frac{1}{2S}\left(V_{int}(0) + \sum_{k'} V_{int}(k-k') b_{k'}^\dagger b_{k'}\right)\right] b_k^\dagger b_k|\rangle. \tag{2.59}$$

The new energy for a given k-state is then

$$E(k) = \hbar\omega(k) - \frac{1}{2S}\left(V_{int}(0) + \sum_{k'} V_{int}(k-k')\langle|b^{\dagger}_{k'}b_{k'}|\rangle\right). \tag{2.60}$$

Converting the sum to an integral results in a first order change in energy,

$$\Delta E^{(1)}(\vec{k}) = -\frac{V_{int}(0)}{2S} + \frac{1}{4\pi}\int dk' k' V_{int}(|\vec{k}-\vec{k'}|)\langle|n(k')|\rangle, \tag{2.61}$$

where $n(k') = b^{\dagger}_{k'}b_{k'}$ is the occupation number, and the integration over θ' resulted in a factor of 2π. The details of $V_{int}(k-k')$ are covered by Ciuti et al. [25] This $V_{int}(k-k')$ is the same interaction used for the polariton-polariton scattering. The model for energy corrections does not use this quantity as a constant as was done for polariton-polariton scattering, but as $C\exp(\frac{-q}{q_o})$. C is a parameter that models the strength of the interaction and q_o models how quickly the interaction strength weakens as $q = |\vec{k}-\vec{k'}|$ increases.

2.2.2 SECOND-ORDER ENERGY CORRECTION

Second-order perturbation theory gives,

$$E' = E + \langle i|V|i\rangle + \sum_{m\neq i}\frac{|\langle m|V|i\rangle|^2}{E_i - E_m}. \tag{2.62}$$

Here E is the unperturbed energy and $\langle i|V|i\rangle$ is the interaction described in the previous section.

The second order term is more complicated because the intermediate state, $\langle m|$, does not have to be energy conserving. Nevertheless, converting the sum to an integral results in the second order energy correction,

$$\Delta E^{(2)} = V_{int}n_k \int d\theta_q \frac{dE_q}{2\frac{\partial E}{\partial k^2}|_q}\int d\theta_{k'}\frac{dE_{k'}}{2\frac{\partial E}{\partial k^2}|_{k'}}\frac{C^2 exp(\frac{-2q^2}{q_o^2})(1+n_{k''})(1+n_{k'''})n_{k'}}{E_{k'''}+E_{k''}-E_{k'}-E_k}. \tag{2.63}$$

2.2.3 PHASE-SPACE FILLING

As the density of particles increases, the excited states in the sample begin to fill up. The fermionic constituents of the polaritons begin to have trouble being created. As a result, the excitons start to decouple from the photons. Experimentally, this is seen as the polariton splitting becoming weaker. This leads to a blue shift of the lower polariton line and a red shift of the upper polariton line, i.e. a closing of the line splitting. We deduce from experiment the magnitude of this effect. To fit the data, a parameter is found to change the Rabi splitting in the model as the simulated particle density increases.

3.0 REVIEW OF OTHER KINETIC MODELS FOR MICROCAVITY POLARITONS

Numerical solutions to the Boltzmann Equation have been done for some time now. In general, the equation to be solved has the form,

$$\frac{\partial n_{\vec{k}}}{\partial t} = P_{\vec{k}}(t) + \Gamma_{\vec{k}}(t) + \sum_i W^{(i)}_{\vec{k}\to\vec{k}'}(t). \tag{3.1}$$

Here n is the occupation for the state $\vec{k}$, P and Γ are pumping and recombination/loss terms respectively, and the $W^{(i)\prime}s$ are whatever kind of interaction integrals are to be considered. Using this form, a solution for elastic scattering in bulk material was found by Snoke and Wolfe[36] and Snoke [35]. They analytically reduced the interaction integral down as far as possible, as was done in chapter 2, and then used a computer to numerically calculate that which remained. This significantly reduced the amount of time required for the calculation. Numerous groups have applied the same process to microcavity polaritons with various approximations being made. This chapter will review their publications.

3.1 LOW-DENSITY STUDIES

The early numerical simulations on microcavity polaritons had emphasis on studying the behavior of the low density system in order to understand the interplay between the relaxation rates and the recombination rates. Tassone, *et al.,*[37] used a model with only exciton(polariton)-phonon scattering and polariton recombination(photoluminescence), where they modeled the behavior of the distributed Bragg reflector exactly. $P(t)$, the pumping term, was an instantaneous pulse at the beginning of the simulation on the excitonic

part of the lower polariton branch. Their interest was (1) in the recombination rate given by $\Gamma(t)$ as a function of emission angle, and (2) in the photoluminescence rise and decay times as the temperature and detuning were varied.

They found that the dependence on detuning of the radiative rate was determined by the composition of the polariton being considered. As the polariton became more photonic, the radiative rate increased, and the opposite occurred as the polariton became more excitonic. They also concluded that photoluminescence decay times increased with temperature, while rise times decreased. Neither were significantly affected by detunings of up to $2\Omega_R$. These two times are mostly a function of the dynamics occurring in the excitonic region of the dispersion curve. Their calculations indicated that the excitonic region of the lower polariton was nearly thermalized, but that relaxation to the lower energy states may be inhibited, a so-called bottleneck. This is similar to what is observed in the dynamics of the bulk.

The possible existence of a bottleneck led Tassone, *et al.,* [12] to numerically study that phenomenon next. A full calculation for the Bragg reflectors was again taken into account. Instead of an instantaneous pulse of excitons on the lower polariton branch, this study included a non-resonant pumping term. They calculated the formation rate of the lower polaritons as a function of polariton energy while taking into account acoustical scattering and optical phonon emission (see Figure 3.1). The simulations discussed in Chapter 5 of this thesis assume the pumping rate into all states to be a constant. Tassone's simulations remained in the low density regime and they continued to only consider phonon scattering. They concluded that indeed there is a bottleneck effect in the microcavity polariton dynamics.

The bottleneck was an obvious barrier to the thermalization of the polaritons. A way to overcome the barrier was needed. Exciton-exciton scattering rates become important at higher densities (scattering rate is proportional to n). This was first numerically considered by Tassone and Yamamoto [26]. In this study, exciton-exciton scattering rates were considered along with the exciton-phonon scattering rates. Additionally, the pumping term was returned to being resonant. They included a term that accounts for the interaction of the excitons and the strong electromagnetic field due to the pump. Even at high densities their model still showed that polariton-phonon scattering could not overcome the bottleneck. The

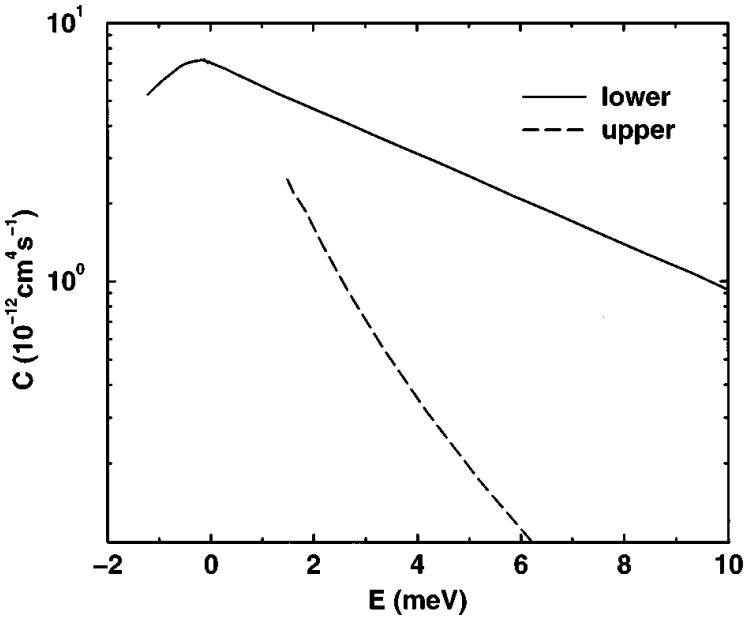

Figure 3.1: From [12], the calculated formation coefficient, C, for the equation $F(E) = Cn_c^2(E)$ where $F(E)$ is formation rate of upper and lower polaritons for a non-resonant pump and $n_c(E)$ is the carrier density. $E = 0$ is the bare exciton energy.

peak occupation occurs at the lower energies of the uncoupled excitonic energies (see Figure 3.2) and polariton occupations decrease as their energies decrease without thermalizing. Only with a pumping term into the lower polariton branch could large populations below the bottleneck be achieved.

Tassone's and Yamamoto's publication provides useful insight for those researchers doing a numerical simulation. They describe four criteria important for determining the step size

for the energy bins. These are:

1. $\Delta E < k_B T$. The distribution functions, which at low density are proportional to $\exp(-E/k_B T)$, are best approximated by small steps of energy.

2. $\Delta E < \hbar c \bar{q}$. $\bar{q}$ is the average phonon momentum exchanged. In GaAs quantum wells this energy is near 1 meV.

3. Good results with changes in ΔE. They found energy steps between 0.05 meV and 0.4 meV gave consistent results.

4. $\Delta E > \hbar \Gamma$. Γ is the exciton-exciton scattering rate. ΔE represents an uncertainty in E. The larger ΔE is, the smaller the uncertainty in Δt, the time step for each iteration. This Δt needs to be much smaller than the time it takes for the average particle to scatter. This is because Fermi's Golden Rule assumes the scattering process is quick compared to the time between scatterings so that no information of the scattering is retained. They report energy steps of 0.1 meV as adequate for exciton-exciton scattering with density, $n_{exc} \sim 10^9 cm^{-2}$.

3.2 HIGH-DENSITY STUDIES

Numerous groups have used the Boltzmann equation to explore polariton condensation and polariton lasers. To overcome the bottleneck, without pumping directly into the lower polariton region, higher densities were needed. Experimentally, high density can lead to problems with the polaritons losing their bosonic behavior and this should be kept in mind when considering numerical results.

As the density increases, one mechanism that becomes important is polariton-polariton scattering. Polariton-phonon scattering rates are proportional to the density of phonons, that is, there is a dependence on the lattice termperature. Polariton-polariton scattering rates are proportional to the density of polaritons squared. This type of scattering can become significant, in some samples, at densities before the polaritons lose their bosonic behavior. Various papers have considered this term and made other approximations to study the kinetics.

Malpuech, *et al.*,[38] explored free electron-polariton scattering in addition to polariton-

polariton and polariton-phonon scattering. Since free electrons rapidly cool, the polaritons near the bottleneck could effectively lose energy. Additionally, the dipole-charge scattering matrix element is larger than that for dipole-dipole[39],[40]. With this mechanism included they were able to show large populations in the lower polaritons near $k = 0$.

In a series of papers [41, 42, 43, 44], Doan, *et al.*, showed the possibility of large accumulations in the ground state. The first paper [41] showed with the correct choice of parameters that it was theoretically possible for acoustic phonons to overcome the bottleneck. They chose cavity lifetimes of 50 ps as opposed to typical lifetimes of current samples of about 1 ps. In [43] an approximation was made that the sample thickness was on the order of 10 μm. This causes the first excited polariton state to be at $k_{\|} = 6$ x 10^5 m^{-1}. This wavevector occurs at about 1/20 of the lower polariton wavevector space. In [44] a similar study was done for II-VI materials. With these approaches, this group was able to show a steady state Bose-Einstein distribution could occur.

Porras, *et al.*, [45] followed Tassone and Yamamoto's 1999 model with some simplifications to the numerics. Many models assume that the polaritons are interacting with a phonon bath at constant temperature. The interaction of polaritons with phonons is a quick calculation compared to the time taken to calculate the interaction with lower polaritons and low energy excitons. One way to speed up the calculation is to separate the lower polariton dispersion curve into a polariton region and a thermalized exciton region. Thus, for the exciton region,

$$n_{xi} = n_x e^{-\beta_x \epsilon_{xi},} \quad (3.2)$$

with n_x the occupation for the lowest energy exciton, $\beta = k_B T$, and ϵ_{xi} the energy of excitons with higher energy. In so doing, the populations become disjoint at the bottleneck and this region is neglected.

To further simplify the system they use a quantized area. As in the case of the papers [41, 42, 43, 44], this causes the wavevectors to be discrete. They also do the calculation in k-space instead of doing the calculation in energy space. This makes it easier to have a large number of low energies, the main region of interest, but does not simplify the calculation. In three dimensions the equations can be analytically reduced further in energy space than in k-space. In two dimensions there is an integral that cannot not be simplified which makes

doing the calculation in k-space equivalent to doing the calculation in energy space.

With these methods, they get two coupled equations, one for the lower polaritons,

$$\frac{\partial n_k^{lp}}{\partial t} = W_{ph} + W_{xp} - \Gamma_k^{lp} \tag{3.3}$$

and one for the excitons,

$$\frac{\partial n_k^{x}}{\partial t} = W_{ph} + W_{xp} - \Gamma_k^{x} + P_x. \tag{3.4}$$

P_x is a pumping term that populates the exciton reservior with excitons at the lattice temperature. They assume that the polaritons do no interact since there is no polariton-polariton scattering term.

In a scattering process that moves an exciton to the lower polariton state, another exciton must gain energy. This process heats the excitons. To complete their model they use the following equation,

$$\begin{aligned} \frac{\partial e_x}{\partial t} &= -\frac{1}{S}\sum_k \epsilon_k^{lp} dg_k^{lp} [W_l^{in}\left(\frac{n_x}{S}\right)^2 (1+n_k^{lp}) - W_k^{out}\left(\frac{n_x}{S}\right) n_k^{lp} + \\ & \left(\frac{\Delta E n_x}{k_B T_x S}\right) \sum_i \epsilon_i^x \sum_j W^{ph} T_L e^{-\beta_x \epsilon_i^x} + p_x - \Gamma_x. \end{aligned} \tag{3.5}$$

where e_x is the energy in the exciton reservior. There is only this term for the reservior, where the sums are over the lower polariton k-space. This one term replaces a large number of bins in the excitonic k-space and reduces the time demand for the calculation.

They were able to show that their model reproduces the calculations of Tassone and Yamamoto[26] at low densities. They were also able to show, using values consistent with CdTe, that large occupation numbers in the lower polariton states could be achieved with pumping levels as high as 1.5 x 10^{11} cm^{-2}. This density is 20 times more than what Tassone and Yamamoto had used but is less than Porras's calculated saturation density of 6.7 x 10^{11} cm^{-2}[45]. Figure 3.3 shows their published calculations.

Chaves and Rodriquez [46] included polaritons scattering with free electrons as well as scattering with acoustical phonons. They ignored polariton-polariton scattering mechanisms where Malpuech *et al.*[34] had included them in their study. To obtain free electrons an experimental sample would have to be doped. They, too, were able to show that transition

beyond the bottleneck region was possible. They did not provide the details of the polariton-electron scattering matrix element.

Some of the most elaborate models have started with a condensed phase already present and then take into account a Bogoliubov transformation[47], [48]. Sarchi and Savona's model follows along a path given for atomic condensates discussed in [49], [50]. In these studies not only do they keep track of occupation numbers, but allow the dispersion relationship to shift in the energy due to particle-particle interactions. The previous chapter discussed how many body interactions cause energy shifts. Their method breaks the population up into three regions: a condensed polariton region, an excited polariton region, and an uncoupled exciton region.

There are three equations that model the occupation of the states, one for each region: the condensate

$$\frac{\partial n_c}{\partial t} = \gamma_0 n_c + \frac{\partial n_c}{\partial t}|_{ph} + \frac{\partial n_c}{\partial t}|_{XX} + \frac{2}{\hbar}\sum v^{(k)}_{k,-k}\mathrm{Im}(\tilde{m}_k) \tag{3.6}$$

the excited polaritons

$$\frac{\partial n_k}{\partial t} = \gamma_0 n_k + \frac{\partial n_k}{\partial t}|_{ph} + \frac{\partial n_k}{\partial t}|_{XX} + \frac{2}{\hbar}\sum v^{(k)}_{k,-k}\mathrm{Im}(\tilde{m}_k) \tag{3.7}$$

and the uncoupled excitons,

$$\frac{\partial n_x}{\partial t} = -\gamma_x n_x + \frac{\partial n_x}{\partial t}|_{ph} + \frac{\partial n_x}{\partial t}|_{XX} + f. \tag{3.8}$$

Here $\gamma' s$ are inverse particle lifetimes, $v' s$ are polariton-polariton interaction terms, and f is an incoherent pumping term into the excitons. ph and XX stand for interactions with phonons and excitons respectively. These terms have been discussed in Chapter 2. The lower polariton parts have a new term with $\tilde{m}$, the scattering amplitude. It is given by

$$\frac{\partial \tilde{m}}{\partial t} = -2\left[\gamma_0 + i\omega_k + \frac{i}{\hbar}v_{k,0}^{(0)}(n_c - n_k - 5/2)\right]\tilde{m}_k -$$

$$\frac{i}{\hbar}\left[\sum_q v_{q,-q}^{(k-q)}\tilde{m}_q - 2v_{k,-k}^{(k)}n_c(n_c-1)\right](1+2n_k) +$$

$$2\frac{i}{\hbar}\Upsilon(n)\Big\{v_{k,-k}^{(k)}\left[2\chi_k\bar{n}_k\left(\chi_k\bar{n}_k + 2|V_k|^2\right) + 2|V_k|^4\right] +$$

$$U_kV_k^*(1+2\bar{n}_k)\sum_k v_{k,-k}^{(q)}U_q^*V_q(1+2\bar{n}_q)\Big\}. \tag{3.9}$$

Here

$$|V_k|^2 = \xi_k\frac{[E_k - (\omega_k + v_{k,0}^{(0)}\xi_k)]^2}{(v_{k,k}^{(k)}\xi_k)_k^2 - [E_k - (\omega_k + v_{k,0}^{(0)}\xi_k]^2}, \tag{3.10}$$

$$\Upsilon(n) = n^2\frac{1+2n_c}{(n_c+1)(n_c+2)}, \tag{3.11}$$

$$\chi_k = \xi_k + 2|V_k|^2, \tag{3.12}$$

$$|U_k|^2 + |V_k|^2 = \xi_k, \tag{3.13}$$

and

$$\xi_k = \frac{n_c + n_k}{n}. \tag{3.14}$$

The total number of particles is given by, n.

They simultaneously solve these equations. Figure 3.4 shows the results of their calculation. The lower curve in the plot is from an analytical equilibrium calculation. The break in the upper plots are because they have continued to ignore effects due to the bottleneck region.

Using GaAs parameters, quantization lengths of between $10\mu m$ and $30\mu m$, and a Rabi splitting of around 7 meV, their main conclusion was that cavities need to be designed with

only moderately longer polariton lifetimes, $\tau_{pol} \sim 10$ ps, to give the possiblility of studying thermalized distributions.

All of the reports on numerical simulations in this section have shown that the $k = 0$ state can have orders of magnitude more population than more energetic polaritons below the bottleneck. None of these groups have compared numerically simulated distribution functions to experimental data.

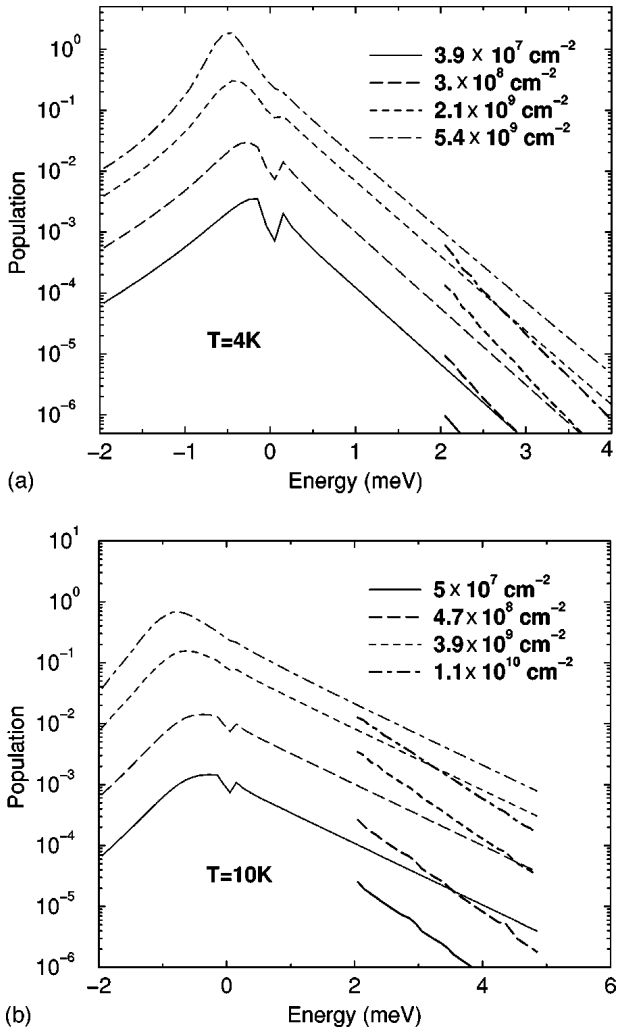

Figure 3.2: Occupation number vs energy for polaritons. The polariton density for each simulation is given in the upper right hand corner of each graph. The existence of the bottleneck, the peak in the curve, remains when polariton-polariton scattering is considered along with polariton-phonon scattering from [26]. $E = 0$ is the bare exciton energy. The bottleneck is pushed to lower energies with higher density, but never goes to the lowest energy.

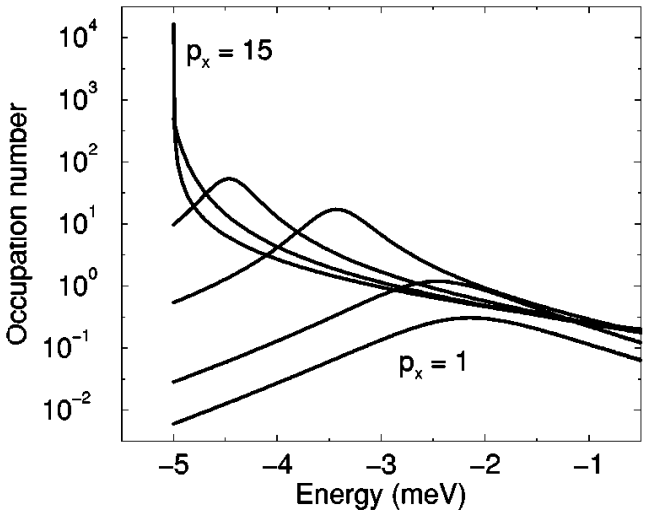

Figure 3.3: Porras, *et al.* [45], showed that numerical simulation suggested the possibility that strongly pumping a material like CdTe, with higher saturation density than GaAs, would result in a large occupation of the lowest energy states. Their pumping density was less than, but on the same order of magnitude, as the saturation density for CdTe. P_x is the pumping rate into the system in $cm^{-2}/100$ ps. P_x is shown for 1, 2, 5, 8, and 15. Notice that the distribution is sloped, not flat, for energies below the bottleneck.

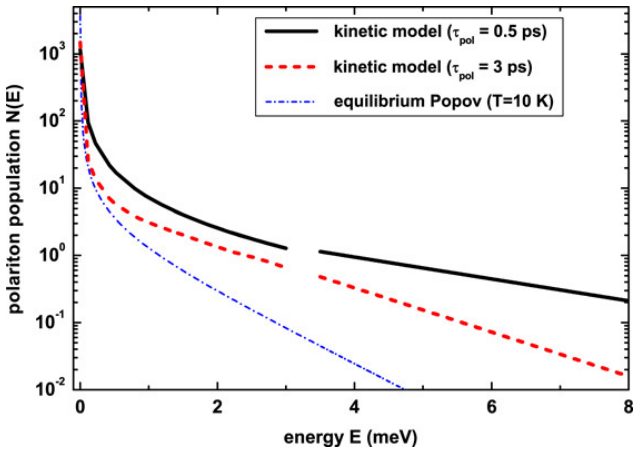

Figure 3.4: Numerical simulation of polaritons from [48]. GaAs parameters for the effective masses, deformation potential to acoustic phonons, and Coulombic and Pauli exclusion terms.

4.0 EXPERIMENTS

This thesis provides data from two main experiments. The first is angle-resolved luminescence from the microcavity. Data for this experiment was taken using CW and quasi-CW pumping conditions. The second experiment is time resolved luminescence from the ground state of the microcavity under pulse pumping conditions. Data from the angle-resolved experiments is compared to numerical simulation in the next chapter.

4.1 ANGLE-RESOLVED MEASUREMENTS

Figure 4.1 is a picture of a set up near one of the cryostats used. The lower portion of the cryostat is visible as well as some of the opto-mechanics used in its vicinity. The arrows (red when in color) drawn in the picture represent the path of the pump beam. The metallic arms in the foreground of the picture meet underneath the cryostat. The center of their rotation is placed to coincide directly underneath the front surface of the microcavity based on an optical path. The arms are constructed so that they can be moved past each other. In this way, the sample can be pumped from various angles and the luminescence can be collected from various angles.

Once the microcavity is cooled to liquid helium temperatures, preparation for angle-resolved experiments is made by imaging reflected white light from the sample onto the front slit of the spectrometer. The spectrometer is tuned to near the edge of the stop band for the microcavity. Figure 1.9, between the energies of 1.7 eV and 1.75 eV, provides an example of what the spectrometer would show. Weak laser light is then scattered off the sample which shows up as a dim line on the spectrometer. The laser is tuned to the

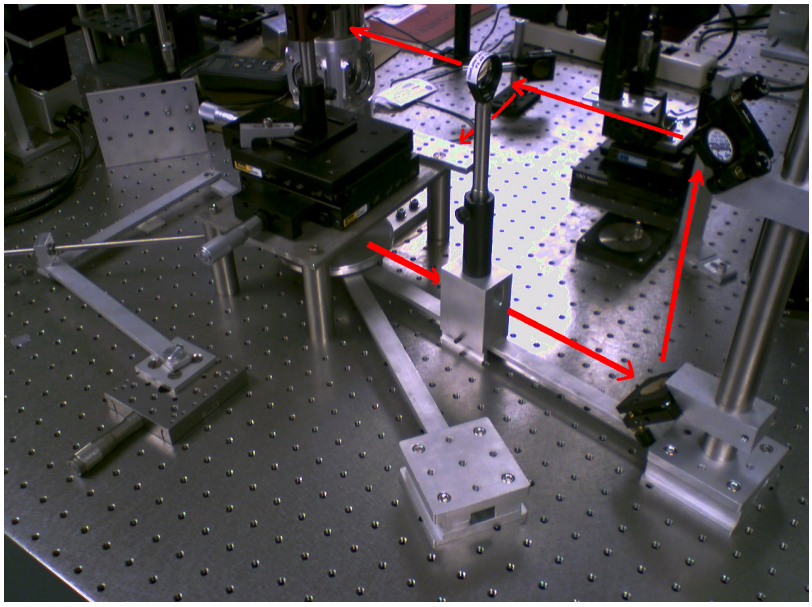

Figure 4.1: Optical set up near the cryostat. At the time this picture was taken the cryostat had been replaced by a mirror. The cryostat sits in the background.

wavelength of the sample's first minimum reflectivity above the stop band. Pumping this region excites free electrons and holes well above the energies of the exciton, which is 1.59 eV. These particles must interact with several optical phonons, 37 meV, to relax to the exciton states. Interacting with the optical phonons ensures that coherence from the laser pump is not preserved in the exciton states. After a time of less than a nanosecond, the polariton distributions reach a steady state.

The pump laser is incident on the sample in a region where the excitons are detuned from the cavity mode. The luminescence from the sample is imaged onto the spectrometer, which has now been tuned to the cavity energy, around 1.6 eV. A bright cavity mode is seen on the CCD image. Stress is then applied to the sample via the pin, which was described in the cryostat section of Chapter 1. As the stress is increased, the exciton mode begins to

come into resonance with the cavity mode. A spatial dip in the polariton mode creates our trap as shown in Figure 1.7.

Polaritons decaying into external photons are measured as luminescence from the microcavity. As stated in the introduction, each momentum state of the polaritons emits photons at its own specific angle to the normal of the microcavity surface. A fiber optic collects this luminescence at a given angle. This luminescence through the fiber optic is analyzed with the spectrometer. The ouput of the spectrometer is collected by a CCD that integrates over time. This measurement is repeated by moving the fiber optic through a total arc from $-19°$ to $19°$ in $1°$ steps. Aggregated images of the data by angle are shown for each pump power, 1 mW in Figure 4.2, 6 mW in Figure 4.3, 24 mW in Figure 4.4, 35 mW in Figure 4.5, and 80 mW in Figure 4.6. The CCD's spatial information has been integrated over since all spatial information is lost going through the fiber optic. These aggregated images show the dispersion relationship of the lower polaritons.

There are two effects to increased pumping power that are noticable between Figure 4.2 and Figure 4.6. The first is that as the density of polaritons increases, the population becomes concentrated near $\vec{k} = 0$, which occurs at an angle of $0°$. While this is good, we also see that the dispersion curves become distorted. There is a blue shift related to the exciton component of the polaritons interacting with other polaritons. At high powers, > 80 mW, the $\vec{k} = 0$ state is no longer the lowest energy state and the polaritons drift into higher momentum states.

We found while experimentally investigating the polariton dynamics, as shown in Figure 4.7, that there are thermal effects from using a CW laser that take several seconds to stabilize. These thermal effects become apparent when pumping the side of the stress well. The system should reach steady state in much less time than the integration time for each frame. In steady state the luminescence should be nearly constant, but it takes three seconds to reach a constant luminescence. Also, as the sample heats up from having a laser focused onto it, the drift speed of the polaritons increases, allowing them to travel farther within the integration time. An explanation may be that it takes time for the heated lattice to come to steady state with the helium gas that is cooling the sample. To limit the heating of the sample, an acousto-optic modulator was introduced into the set up. This chopped the CW beam

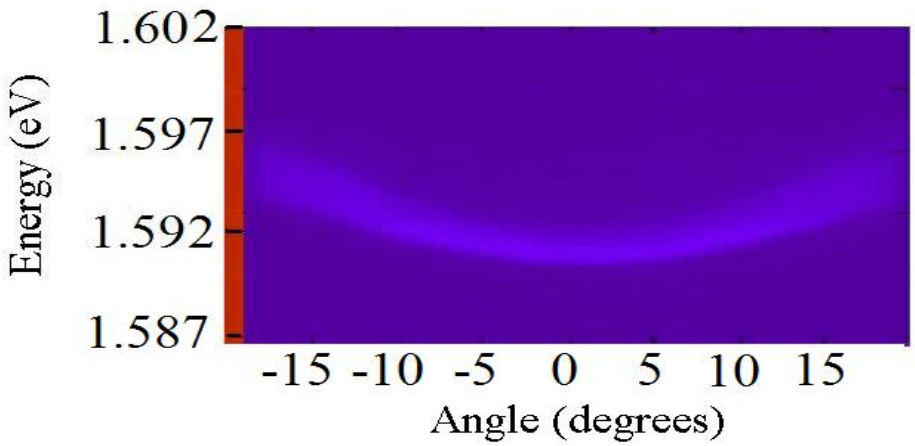

Figure 4.2: A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 1 mW of incident pump power.

and provided pulses of 24 μs at 1 kHz repetition. Repeating the experiment of pumping the side of the stress well showed that luminescence intensities were constant immediately at the beginning of pumping. The drift of the polaritons was also constant. This pulse duration of 24 μs is still hundreds of time longer than the time it takes for the polaritons to reach steady state from initial excitation. We continue then to integrate over at least 300 of these steady state emissions for each angle. Aggregated data similar to Figures 4.2-4.6 are shown in Figures 4.8-4.12.

Figures 4.2-4.6 and Figures 4.8-4.12 can be analyzed to obtain a quantity proportional

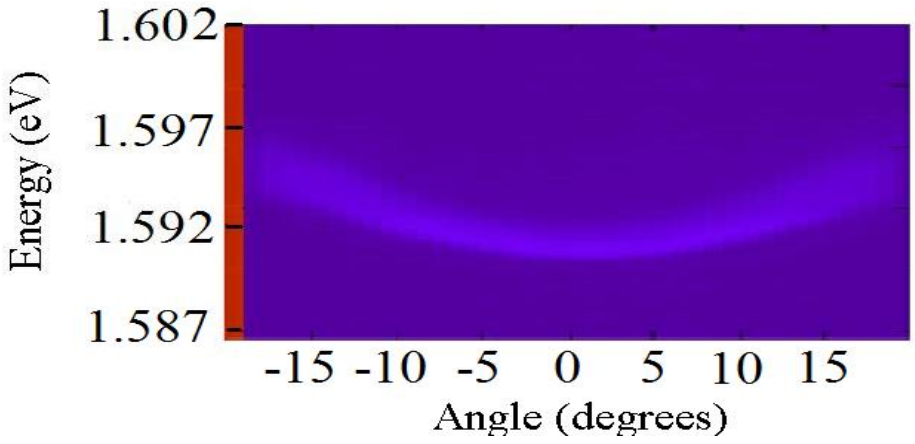

Figure 4.3: A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 6 mW of incident pump power.

to the occupation of each state. First, for each angle the peak energy can be determined. Second, the total intensity at each angle is determined and adjusted for any differences in integration time. This integrated intensity for each angle is related to the occupation of the corresponding state. The emission is inversely proportional to the lifetime of that state and proportional to the state's occupation number. That is, a state that has half the lifetime of another state but equal population will emit twice as strongly over a given time. The integrated intensity is adjusted for the relative lifetime of our states as follows:

The lifetime of a state, τ_k, is given by [51]

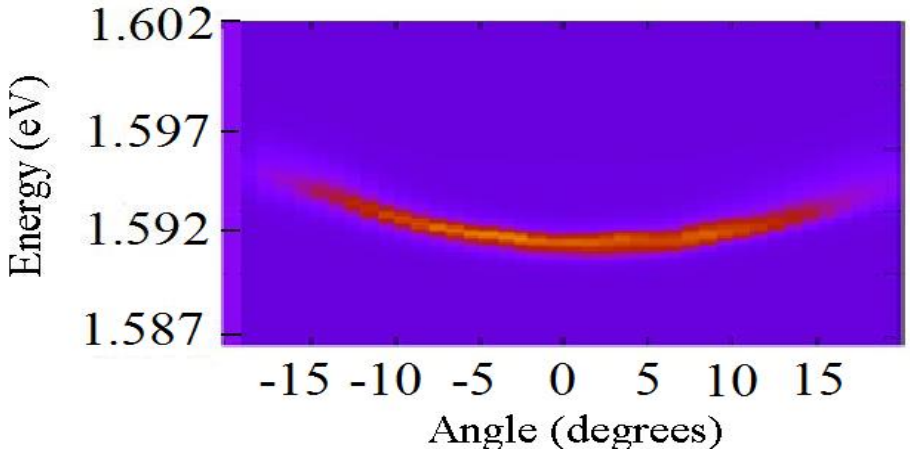

Figure 4.4: A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 24 mW of incident pump power.

$$\frac{1}{\tau_k} = \frac{X_k^2}{\tau_x} + \frac{C_k^2}{\tau_c}. \tag{4.1}$$

Here, X_k^2 is the fraction of the polariton at state k that is exciton and C_k^2 is the fraction of the polariton that is photon (see Equations 1.21 and 1.22). τ_x and τ_c are the lifetimes of the uncoupled excitons and photons in the cavity. The exciton lifetime is substantially longer than the cavity mode so it can be discounted from the equation. The fraction of the lower polariton that is photonic is given by the square of the Hoppfield coefficient,

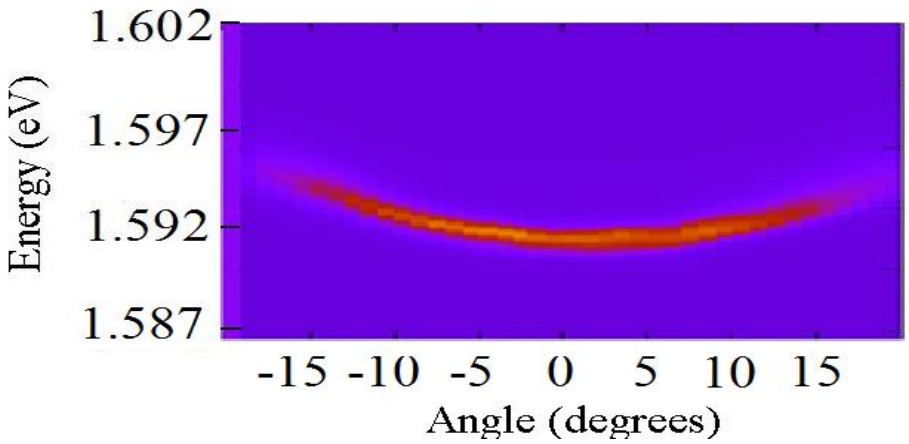

Figure 4.5: A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 35 mW of incident pump power.

$$C_k^2 = \frac{1}{2}\left(1 - \frac{E_{ck} - E_{xk}}{\sqrt{(E_{xk} - E_{ck})^2 + (2\hbar\Omega)^2}}\right). \tag{4.2}$$

To use the preceding equation, the energies of the uncoupled exciton and photon modes need to be known. They can be deduced from either the CW or quasi-CW data. Figure 4.13 shows a plot of the energy versus $k_{\|}$ for the 1 mW CW data. A least-squares calculation of the polariton dispersion, Equation **??** was done by varying the bare exciton energy, bare photon energy, and the Rabi energy, $\hbar\Omega_R$. The solid line in Figure 4.13 shows the result

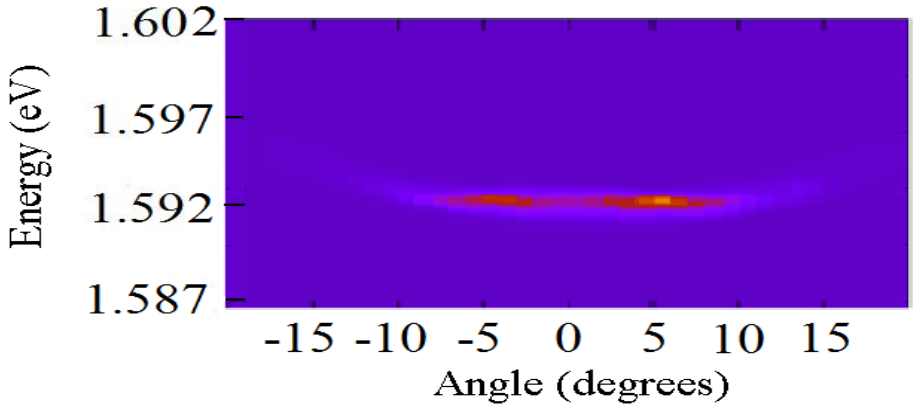

Figure 4.6: A composite of the angular resolved data under CW pumping conditions. For each angle the image on the CCD is integrated over the spatial axis and the intensity is color plotted as a function of energy. This figure is for 80 mW of incident pump power.

of this fit. The best results are for $\hbar\Omega_R = 15$ meV and a detuning of about $E_x - E_c = 1$ meV. Detuning to within 1 meV is within the limit of uncertainty of our experiments. The right-hand side of the plot shows some deviation. An explanation is that even at low pump intensities the excitonic part of the dispersion curve is still very dense. Excitonic interactions may blue shift their energies, pulling the high k-state polaritons energies up. Nevertheless, a small amount of detuning will not affect the relative lifetimes to be calculated.

Having deduced the uncoupled exciton and photon energies, the relative lifetimes as a function of angle are computed. Once these are calculated, a lifetime adjustment is made and

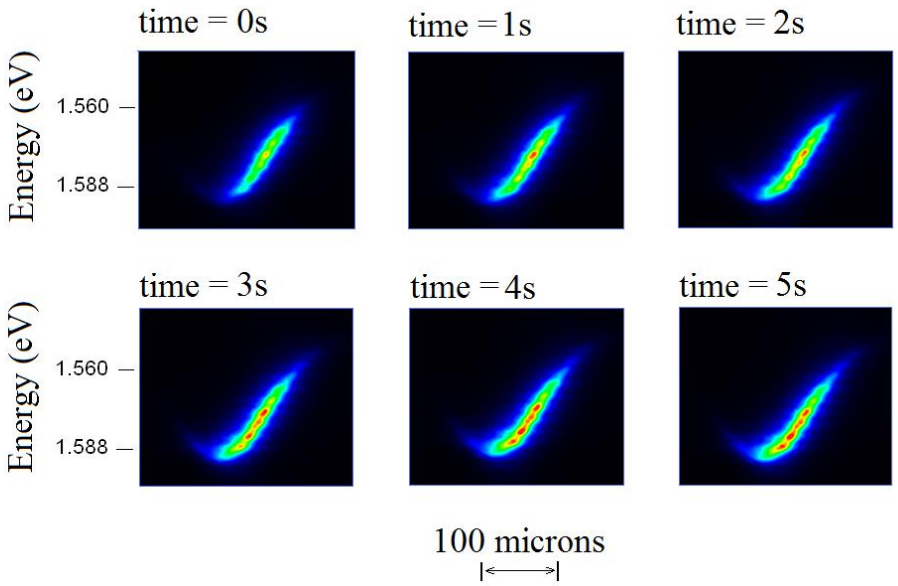

Figure 4.7: CW pumping the side of the stress well. The evolution of the luminescence over five seconds. Thermal effects cause a delay in the build up of the luminescence. The hotter particles also drift farther into the trap since they have a higher average kinetic energy. Each image is integrated over 200 ms. The intensity scale is the same for all images.

plots of the occupation per state versus energy can be made and are shown in Figures 4.14 and 4.15. This data shows the steady-state distribution of the polaritons for the different pumping conditions. They also show attempts to fit the higher density plots with Maxwell-Boltzmann distributions and Bose-Einstein distributions. None of the higher density data are fit well by either distribution. I address this problem in the beginning of the next chapter.

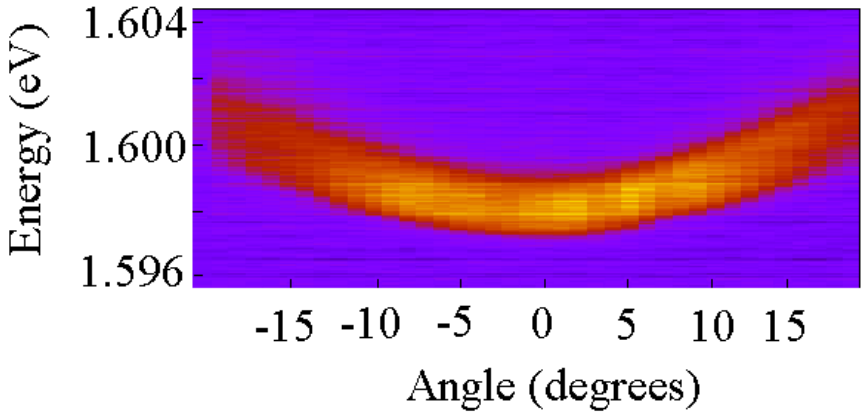

Figure 4.8: A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.05 mW of incident pump power.

4.1.1 THE EFFECT OF STRESS

In the region on the microcavity where the experiments were done, the microcavity is far from resonance without stress. To investigate the effect of detuning, an experiment was performed that varied the stress. Varying the stress varies the detuning because stress shifts the exciton mode relative to the cavity mode. Figure 4.16 shows angle-resolved data of three cases. The top picture is without stress and far from resonance. In this case, the lower polariton is highly photonic. Photons do not interact strongly with themselves and

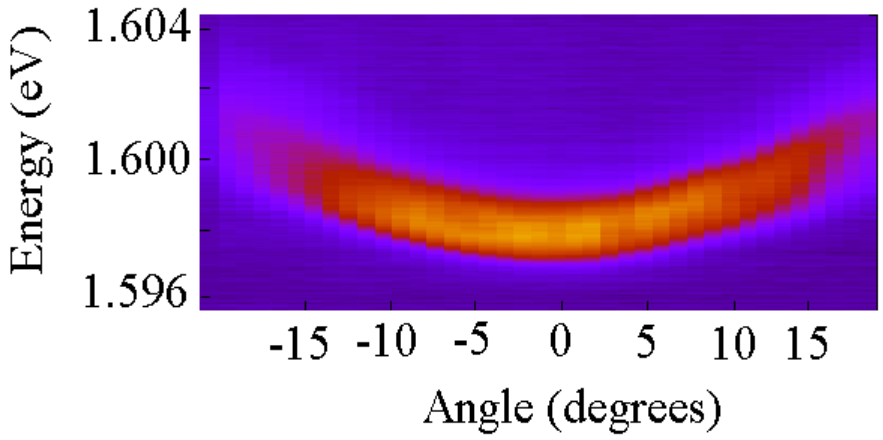

Figure 4.9: A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.2 mW of incident pump power.

the polaritons are trapped in high k states(large angle). Also, due to the steepness of the dispersion curve, phonons are less effective in interacting with the polaritons and allowing transitions to lower k states(small angle, near zero degrees). Stress is increased until, in the bottom picture, the microcavity is in resonance. The lower polaritons are now half excitonic and the dispersion curve is less steep, by 2 meV. Phonons are able to interact better with excitons than with photons. Additionally, the dispersion curve of the phonons is better matched to the dispersion curve of the polaritons, allowing more transitions to the lower k states. The polaritons are able to overcome the bottleneck region of the dispersion curve.

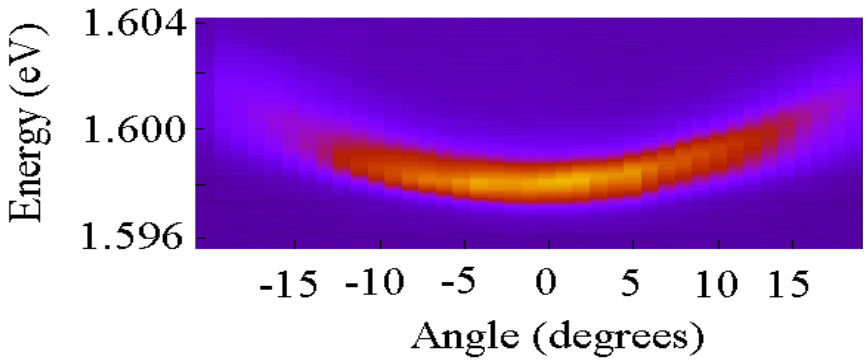

Figure 4.10: A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.4 mW of incident pump power.

4.1.2 LINE BROADENING AND LINE NARROWING

In a noninteracting system the energy of a state is exact. The angle resolved data, Figures 4.2-4.6 and Figures 4.8-4.12, show that for a given angle each emission spectrum has a measurable band of energies. Figure 4.17 shows the spectrum for $k = 0$ when the microcavity is being quasi-CW pumped with 0.05 mW. The linewidth is indicative of the time spent in that state. Using the uncertainty principle, the time that each particle exists in that state is

$$\tau \sim \frac{\hbar}{2\Delta E}. \tag{4.3}$$

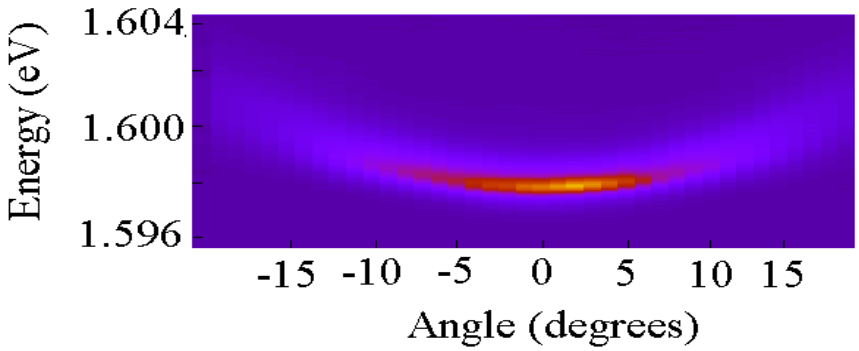

Figure 4.11: A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.6 mW of incident pump power.

For a linewidth of 1.68 meV this corresponds to a time of 200 fs.

The uncertainty principle does not show the full richness of the system. Starting with the uncertainty principle, the linewidth indicates that the particles spend a finite time in that state, but it does not indicate what would be the cause if the linewidth were to change. Figures 4.2-4.6 and Figures 4.8-4.12 show the linewidth narrow as the particle density increases and Figure 4.17 is a plot of the spectral density function when the sample is pumped with quasi-CW light of 0.05 mW. This spectral density function can be modeled as a Lorentzian

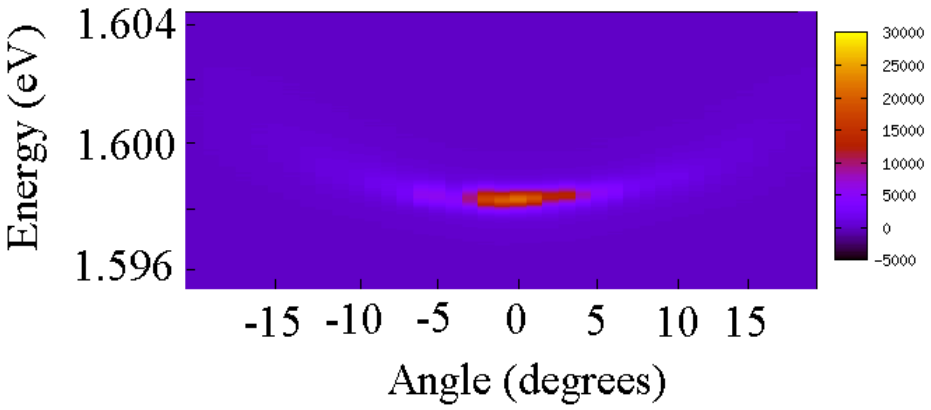

Figure 4.12: A composite of the angular resolved data under quasi-CW pumping conditions. Each angle is spatially integrated and the intensity is color plotted as a function of energy. This figure is for 0.8 mW of incident pump power.

lineshape,

$$S(\omega) = \frac{2\gamma}{(\omega - \omega_o)^2 + \gamma^2}, \tag{4.4}$$

where γ is a damping term that parameterizes the interaction of the system. For N particles, $\gamma \propto 1/N$. As N increases, $S(\omega)$ goes to $\delta(\omega)$, it becomes very narrow. Line narrowing is an indication that a state has become heavily populated [6].

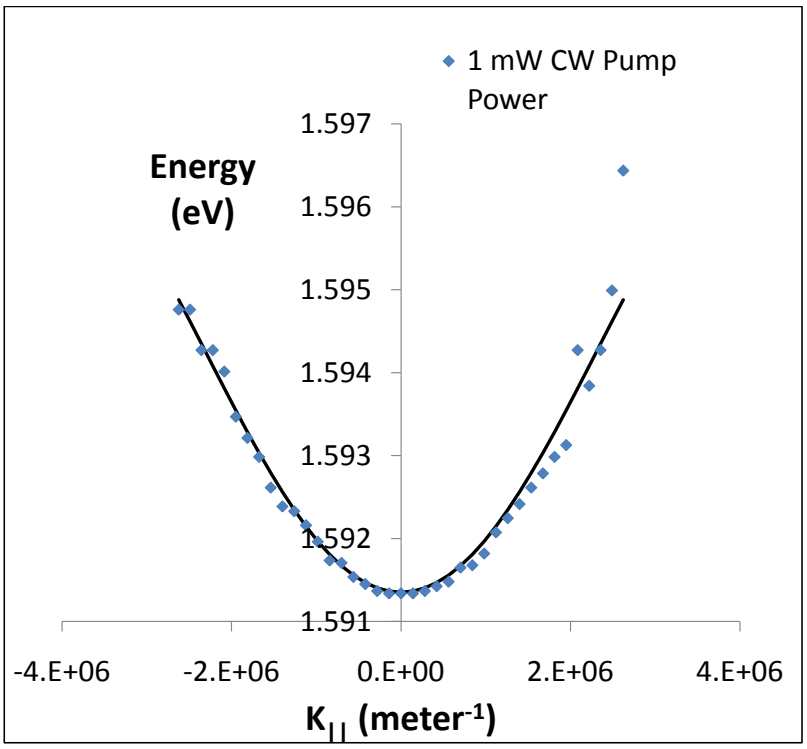

Figure 4.13: Dispersion curve fit to the 1 mW CW laser data shown in Figure 4.2

4.1.3 ERROR ESTIMATES

There are experimental limitations. I discuss those in this section.

The slit width at the entrance to the spectrometer provides some uncertainty in the wavelength. The experiments used a 40 μm slit width which is approximately 3 pixels when imaged on the CCD. Three pixels on the CCD when using the 1800 lines/mm grating in the spectrometer represents an uncertainty in the wavelength, $\Delta\lambda = 0.029$ nm for a wavelength of 775 nm. The percent uncertainty in the photon energy due to the spectrometer/CCD is given by $\Delta E = \Delta\lambda/\lambda = 0.004\%$, or about 0.06 meV. This is less than 1/10 the significant

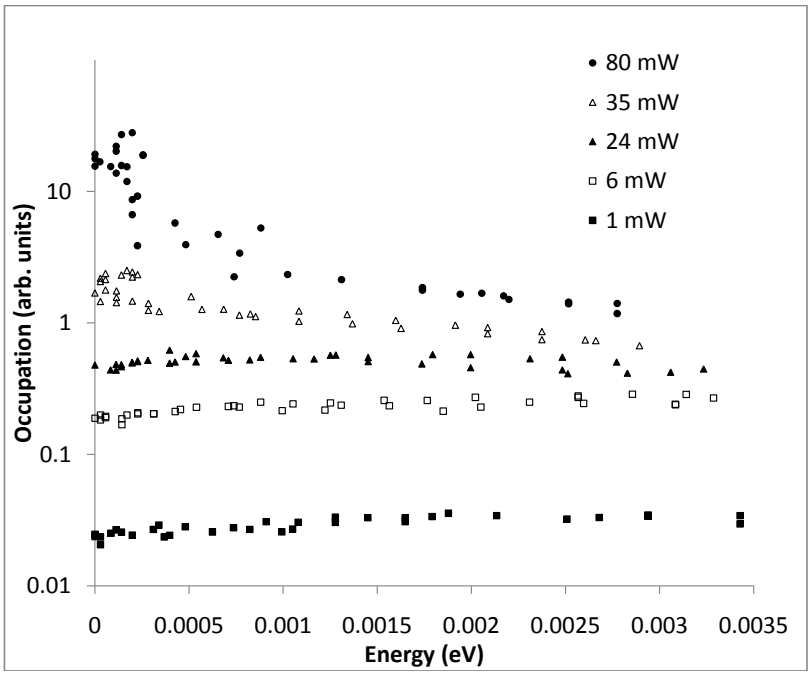

Figure 4.14: Occupation of the lower polariton states for different incident pump powers. These were deduced from the spatially integrated images in Figures 4.2-4.6 for CW pumping conditions.

digit we present in the data.

A large error in the distribution functions could come from the fiber optic if it is collecting from a different number of states at each angle. Since $k_{\|} = k \sin\theta$, then $dk_{\|} = k \cos\theta d\theta$. For the angles in our experiment, $\cos\theta \sim 1$ so that $dk_{\|}$ is essentially constant. At each angle we expect to collect emission from the same number of states. Therefore, we do not make any correction for the collected number of states in Figure 4.14 and Figure 4.15. Additionally,

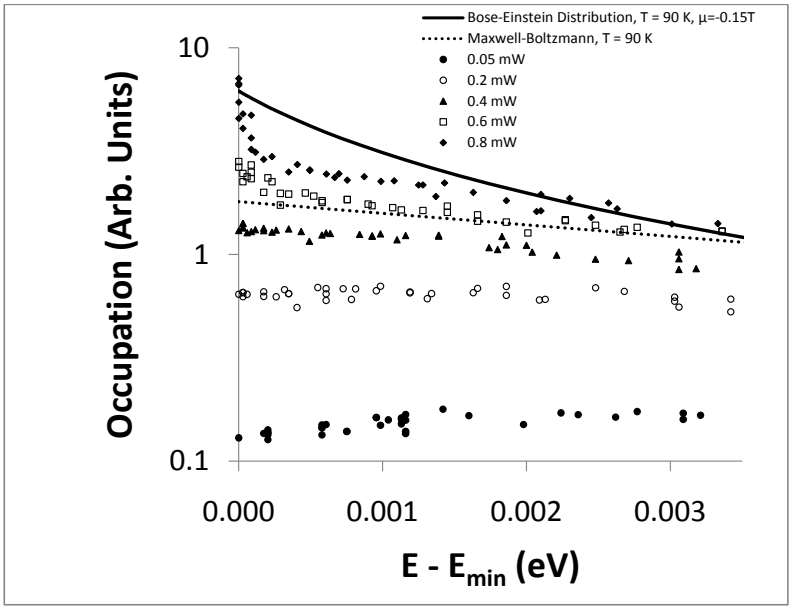

Figure 4.15: Occupation of the lower polariton states for different incident pump powers. These were deduced from the spatially integrated images in Figures 4.8-4.12 for the quasi-CW pumping condition of 2.4% pump duty cycle. The dotted line represents a T = 90 K Maxwell-Boltzmann distribution and the solid line is for a T = 90 K, $\mu = -0.15$kT Bose-Einstein distribution .

the fiber optic aperture has a finite width of 1.6 mm. It collected the luminescence at about 21 cm from the sample. This leads to an uncertainty in the angle of emission from the sample of $\Delta\theta \sim 7$ mrad $= 0.4°$. The angle is then measured to within $\pm 0.2°$. This is 1/5 of the angular spacing between our data points.

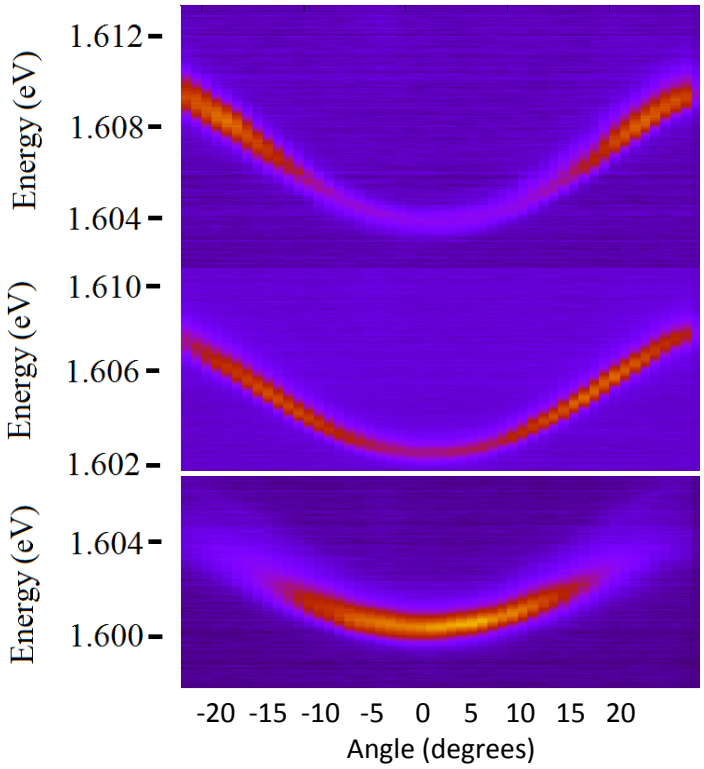

Figure 4.16: The effect of the stress well. From top to bottom the stress is increasing from no stress to resonant stress. Without stress the polaritons are held in high k states. With stress the polaritons are able to make it past the bottle neck. The intensity scales are different for all three plots.

4.2 TIME RESOLVED SPECTROSCOPY

A time resolved spectroscopy (TRS) setup was made that used sum-frequency generation from a BBO crystal. Sum-frequency generation is the process of converting two photons into one photon. The frequency of the new photon is the same as adding the frequencies

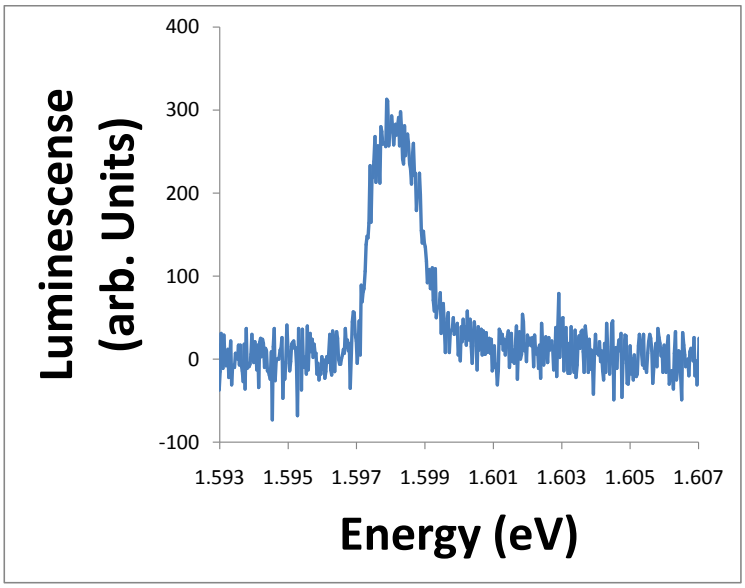

Figure 4.17: The k = 0 spectrum with quasi-CW pumping of 0.05 mW.

of the two destroyed photons. In general, any atom interacting with light will have some portion of that light converted to the sum of the frequencies of photons in the light. If there are many atoms, like in a crystal, the points where the sum-frequency light is generated will be out of phase with light created from other points and will encounter destructive interference. Phase matching is a method where the sum-frequency generated light interacts constructively and becomes amplified. The index of refraction monotonically changes with wavelength for linear optical materials and it is not possible to find a direction to maintain the constructive interference. A birefringent crystal, such as BBO, has ordinary and extra-

ordinary indices of refraction. Paths through the crystal can be found such that the index of refraction for light polarized along the ordinary axis permits the wavefronts to proceed at the same rate as those for light of a different wavelength and different index of refraction along the extra-ordinary axis.

The above described process is most easily described through conserving the energies, $\omega's$, and wavevectors, $\vec{k}'s$. I summarize the result given by Shah[52]. I start with,

$$\omega_1 + \omega_2 = \omega_{sfg} \tag{4.5}$$

and

$$\vec{k}_1 + \vec{k}_2 = \vec{k}_{sfg}. \tag{4.6}$$

where the indices, 1 and 2, refer to incident beams and the index, sfg, refers to the sum-frequency generated signal. This last equation, when considering collinear beams, becomes

$$n_{sfg} = n_1 \frac{\lambda_{sfg}}{\lambda_1} + n_2 \frac{\lambda_{sfg}}{\lambda_2}. \tag{4.7}$$

n is the index of refraction for a given polarization in the crystal and λ is the free space wavelength.

For a uniaxial crystal, the index of refraction for propagation along a direction θ to the optic axis is given by

$$\frac{1}{n^2} = \frac{\sin^2\theta}{n_e^2} + \frac{\cos^2\theta}{n_o^2}. \tag{4.8}$$

Our experiments were done with incident beams polarized parallel to the ordinary axis (subscript o). The angle, θ_{sfg}, necessary to create the sum-frequency generation along the extra-ordrinary axis is found from the preceding equations,

$$\sin^2\theta_{sfg} = \frac{\frac{1}{n_{sfg}^2} - \frac{1}{n_o^2}}{\frac{1}{n_e^2} - \frac{1}{n_o^2}}. \tag{4.9}$$

The indices of refraction for the different beams can be looked up in tables or calculated from a Sellmeier Equation [53].

The preceding equations describe theoretically how the sum frequency generation is accomplished using collinear light. Experimentally, the time-resolved spectroscopy method

does not use collinear incident beams, but the above theory can be used to calculate the approximate angle at which the crystal should be aligned. Beyond this basic calculation the alignment of the crystal is most easily done by trial and optimization.

The sum-frequency generation is used as a gate to allow a signal to be incident on a detector at a specific time. Figure 4.18 gives a simplified experimental set up. In a TRS experiment a short pulse(200 fs) pump beam is split in two by a beamsplitter. One beam continues on to excite the sample. The other beam is sent through a delay stage with a retroreflector. Luminescence from the excited sample is then mixed with the delay pulse in a BBO crystal. The sum frequency generation only occurs when the delayed pulse is incident on the BBO. Thus, the gate only allows a luminescence signal of 200 fs duration through. The intensity of the sum-frequency output is proportional to the luminescence from the sample during the time it is mixed with the delayed pulse.

Figure 4.19 illustrates that only the portion of the microcavity's luminescence that is incident on the BBO at the same time as the laser pulse will create a signal for the detector. Changing the path length of the delay pulse allows different points in time in the photoluminescence to be sampled. The signal from one shot is quite weak but the MIRA is working at 76 MHz. In one second the experiment can be repeated for a given delay position 76 million times. This provides an adequate signal.

An experiment was performed that shows the temporal dynamics of the $\vec{k} = 0$ lower polariton. Figure 4.20 gives the results of the integrated intensity of the sum frequency generated light as a function of time. The pumping wavelength, around 718 nm, was the same as that used in the angle resolved experiments. This pumping method then populates the upper polariton, large k uncoupled exciton states, as well as the lower polariton. The excitonic states and upper polariton states have some transition time to the lower polariton. In effect, these other particles act as a reservoir that can feed the lower polaritonic states as they become photons. For this reason it is reasonable to believe that this measurement does not give the lifetime of the lower polariton. At best it gives some upper bound on the lifetime. From Figure 4.20 this measurement gives a lifetime of 7.7 ps.

The models use the polariton lifetime, and it would be useful to have an experimental measurement of this value. It would reduce the parameter space that is searched when

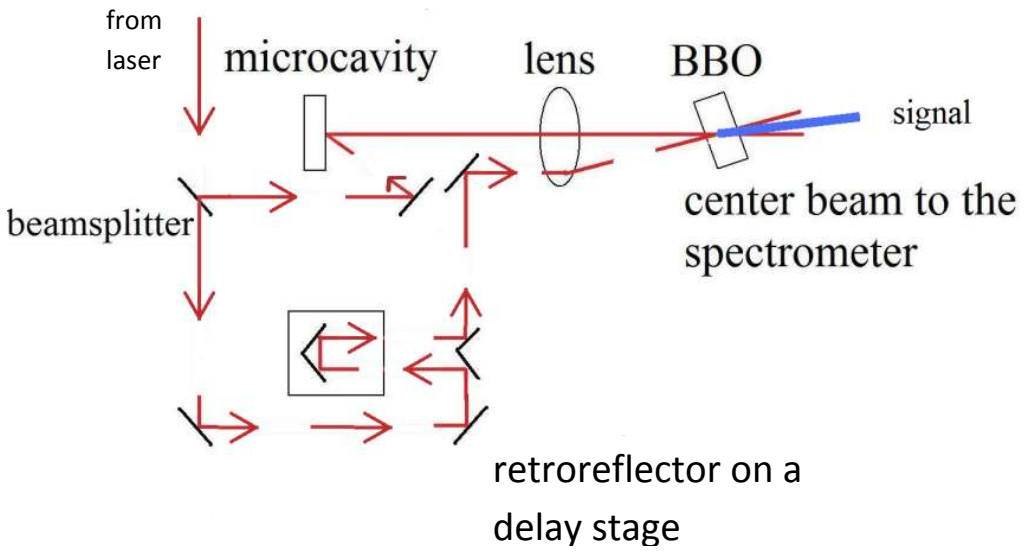

Figure 4.18: A schematic of the time resolved set up. The beamsplitter splits the pump beam. Part of the pump beam is sent to a delay stage. The rest of the pump beam is incident on the microcavity sample. The solid red line from the microcavity represents luminescence.

trying to simulate the steady state results of the previous section. Another method is to pump the lower polariton directly. This method would not excite the upper polariton and exciton reservoirs. Its drawback is that the doubling of the gate pulse is close to the sum-frequency of the luminescence pulse with the gate pulse. The much stronger gate pulse makes it difficult to set the CCD sensitivity to detect the luminescence signal. The results of those experiments have been inconclusive but have suggested a much shorter lifetime. An

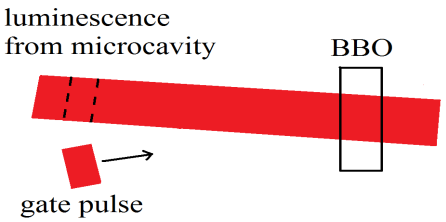

Figure 4.19: The luminescence from the microcavity lasts much longer than the gate pulse. Only a fixed portion of the luminescence mixes inside the BBO with the gate pulse for a given delay. That portion is denoted by the dashed lines. Changing the delay in the gate pulse will sample another point in the microcavity's luminescence.

approximate value for the lifetime will be deduced from the models. They are consistent with the value suggested from Figure 4.20.

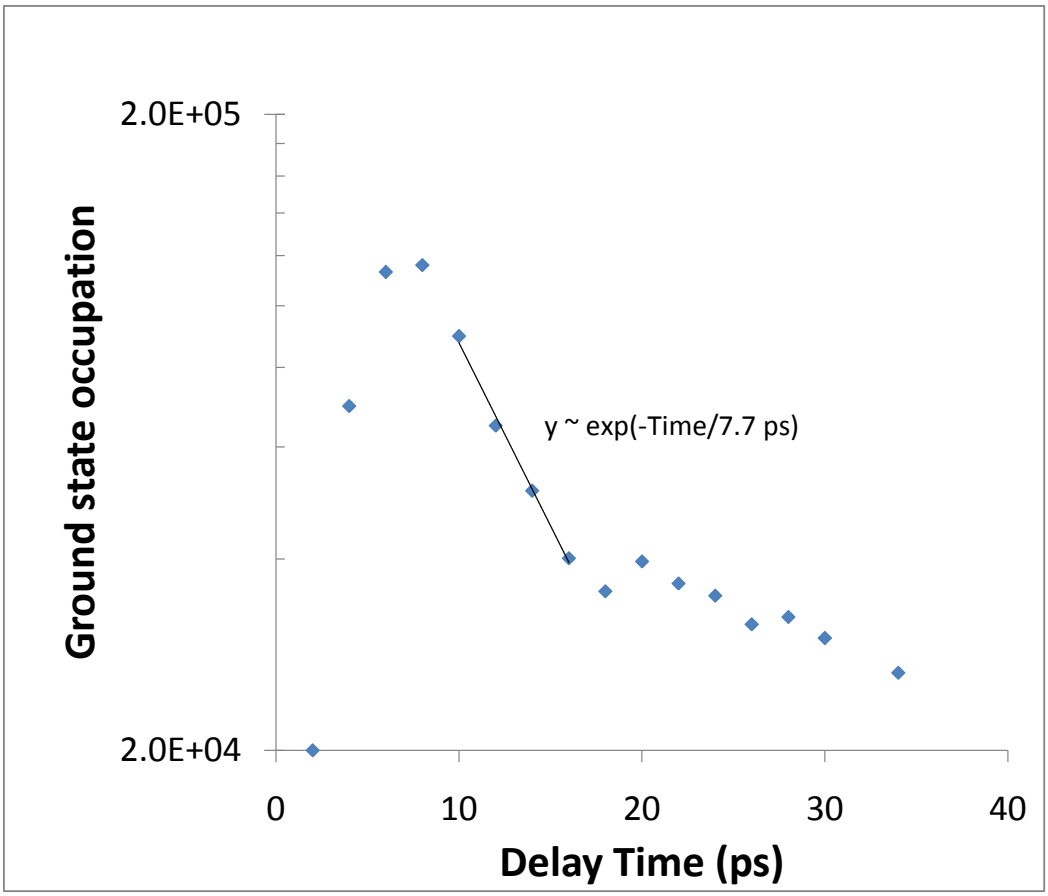

Figure 4.20: The integrated intensity from a resolved spectroscopy of the $\vec{k} = 0$ lower polariton with 141 mW of incident pump power above the stop band.

5.0 NUMERICAL RESULTS

The steady state results provided in the previous chapter are not well fit by Boltzmann statistics or Bose-Einstein statistics. Therefore, the polariton population is not completely thermalized, either to itself or to a bath. Indeed, some of the lower density plots would have a negative temperature. We desire to have an explanation for this result. The scattering rates described in Chapter 2 can be used to simulate, through numerical calculation, the steady state evolution of the polariton population. This chapter describes how the numerical simulation is set up and the results that it provided.

5.1 SIMULATION

To simulate the dynamics of the polaritons we define a mesh in energy space. The mesh is a group of bins; each bin holds the number of particles within the width of that bin for that energy. The whole energy space spans a region up to a point where the highest energies are many times that of the modeled lattice temperature, typically $E_{max} \sim 10k_BT$. This ensures that the highest energy has a very low occupation number, it acts as a boundary condition to control the simulation. We desire to have many points near $E = 0$ and fewer points at higher energies. To accomplish this, the k-space step size for each bin is some multiple of the previous bin's step size. Since the dispersion curve for the polaritons is somewhat flat for low energies, the mesh spacing has very many points near $k = 0$, then becomes sparse near the bottleneck region, and then becomes dense again for the flat exciton region. The number of points is chosen so that the largest ΔE remains below the thermal energy. A plot of bin width versus wavenumber is given in Figure 5.1.

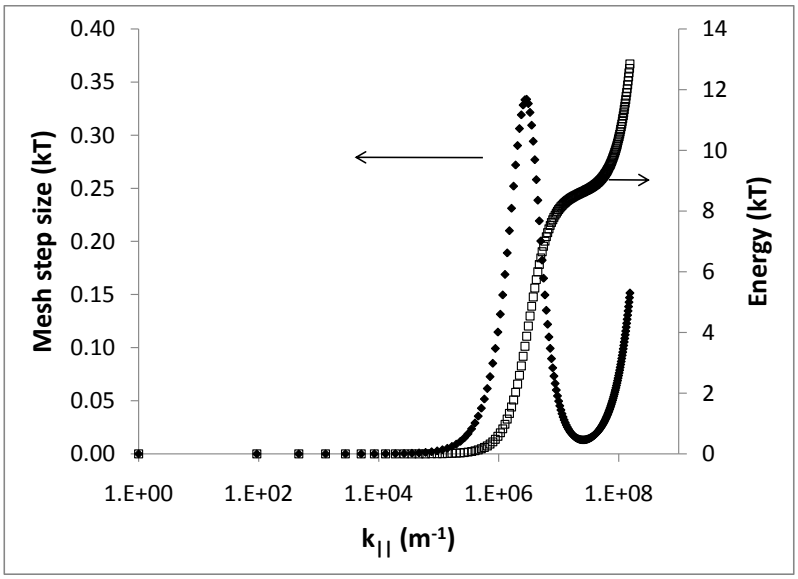

Figure 5.1: The energy step size as a function of bin number on the mesh. The dispersion curve for the polaritons is plotted on the secondary vertical axis.

Once this mesh has been defined, then an average occupation per state at each energy bin is assigned. The assignment depends on the pumping conditions. Our model allows for any length of pulse. A CW laser is modeled by having a pulse longer than the total time simulated. This pulse can have a variety of characteristics. For example, it can model a laser with a gaussian distribution of energies that is pumping the lower polaritons resonantly. In our case, we were simulating the laser being non-resonant, greater than 100 meV above the lower polariton energies. Since the free electrons and holes were created with such high initial

energy, leading to complete randomonization in the relaxation into the polariton states, we assumed that each polariton-exciton state was pumped with equal probability.

After this initial assignment, the simulation calculates the scattering rate in and scattering rate out for each bin on the mesh. In general, the equation is

$$\frac{\partial n_{\vec{k}}}{\partial t} = \sum W_{\vec{k}\to\vec{k}'}(t). \tag{5.1}$$

Here the $W_{\vec{k}\to\vec{k}'}(t)'s$ are obtained from the scattering elements presented in Chapter 2.

The simulation determines the scattering rate for each type of interaction being considered and then linearly combines them. Then the occupation number of each bin is updated. The amount of change is such that the whole system has a certain fraction of particles redistributed. Each bin's change, Δn_k, is proportional to its respective $\partial n_k/\partial t$. The simulation then calculates what time step is needed to move that certain fraction of the population around since

$$\Delta t = \frac{\Delta n_k}{\frac{\partial n_k}{\partial t}}, \tag{5.2}$$

is a reasonably good approximation if Δn_k is small enough. The losses for recombination and the particles added due to the pumping were accounted for using this time step. The overall process is described by,

$$\frac{\partial n_{\vec{k}}}{\partial t} = P_{\vec{k}}(t) + \Gamma_{\vec{k}}(t) + \sum W_{\vec{k}\to\vec{k}'}(t). \tag{5.3}$$

$P_{\vec{k}}(t)$ represents the pumping term and is functionally the same as the term that provides the initial occupation levels. $\Gamma_{\vec{k}}(t)$ represents any loss, and we model it with an exponential decay, exp(-t/τ), where τ is a characteristic lifetime. This process of calculating the changes to n_k continues in a loop, with the new n_k being used to find the new $W_{\vec{k}\to\vec{k}'}(t)$. Figure 5.2 gives an example of how the distribution function evolves over many iterations.

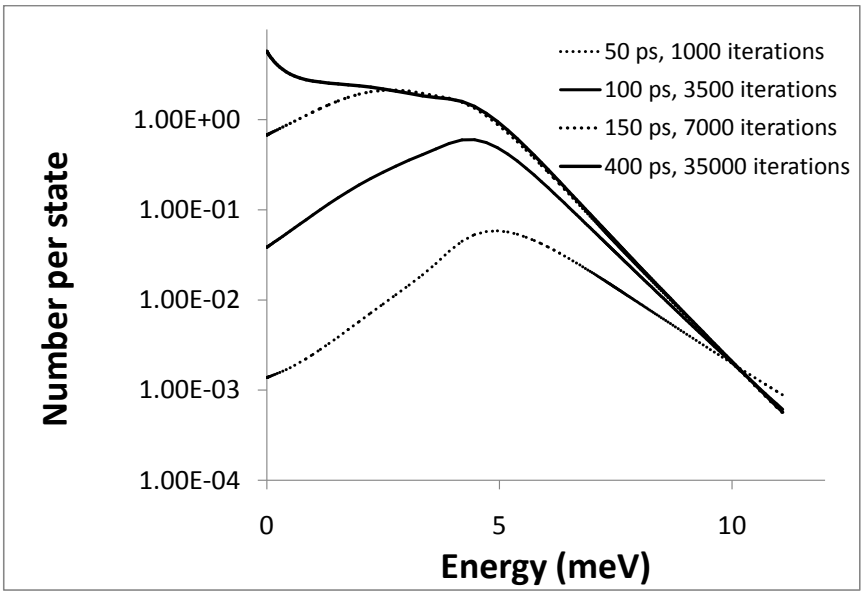

Figure 5.2: The distribution function of the polaritons as it evolved for one set of parameters. Simulated time and the number of iterations are given.

5.2 MODELING THE EXPERIMENTAL DATA

Plots were given in Chapter 4 for angle-resolved data and time-resolved data from the experiments, Figures 4.14, 4.15, and 4.20. The first two are steady state measurements while the third is a time-resolved measurement on the ground state. The simulation can model this data if the correct parameters can be determined. This chapter describes that determination.

The parameter space is rather large and includes a number of variables. Nevertheless,

we can constrain these parameters with experimental input. One is the polariton lifetime, which is related to the cavity mode lifetime and the exciton mode lifetime. Experiments were not able to give a definite value for this. In the early stages of running the simulation, lifetimes for the cavity mode from 100 fs to 20 ps were used. It was found that lifetimes for the cavity mode between 2 ps and 5 ps gave a steady state result that most resembled the data. A cavity mode lifetime of 5 ps was used from then on, which is equivalent to the polariton lifetime being 10 ps. This value is consistent with the upper bound results shown in the time-resolved spectroscopy presented in Figure 4.20. The values of the deformation potentials for the transverse and longitudinal acoustic phonons are given in the literature for GaAs, but they have some uncertainty. The possibility of varying them was investigated but eventually kept at the book value. The lattice temperature, the free electron temperature, the effective polariton-polariton scattering cross-section, the effective polariton-electron scattering cross-section, and the free electron density are other parameters that can be varied in the simulation. The electron temperature was taken to be the same as the lattice temparature. A final parameter is the scaling factor for the experimental data. The simulation gives occupation numbers in absolute values. This scaling parameter is used to shift the experimental data, which is in arbitrary units, to the simulation's results. It must be the same for all the different set of data for different laser powers. The final fits were made by searching the parameter space for lattice temperature, polariton-polariton scattering cross-section, a combined value of polariton-electron cross-section with the free electron density, and the overall data scaling value. The values that were consistent between the CW and quasi-CW cases and for the different polariton densities were found. On one hand there is a large parameter space, but on the other hand, we have a large amount of data at many densities which must be fit, so the parameters were tightly constrained.

5.2.1 INITIAL FIT WITH CHANGING THE EFFECTIVE SCATTERING CROSSECTION

The initial best fits were done using only polariton-polariton and polariton-phonon scattering. It was found by increasing the polariton-polariton scattering rate and adjusting the

lattice temperature that these two scattering mechanisms were sufficient. Figure 5.3 and Figure 5.4 show the results of these fits. The quasi-CW plot has the lattice temperature set at 4 Kelvin for all generation rates. The plot for CW pumping required the lattice temperature to be increased with increasing power. This is to be expected since we know that there is a lot of excess heat created when pumping in CW mode.

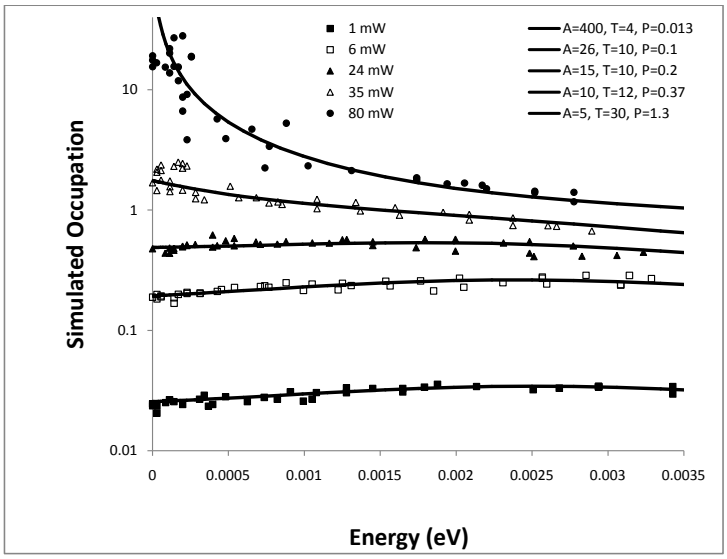

Figure 5.3: A fit to the CW pumped data using polariton-polariton and polariton-phonon scattering. "A" stands for the coefficient used in front of the polariton-polariton scattering cross-section and "P" is the generation rate used. Simulated plots are shown next to their corresponding experimental pump power.

A plot of the magnitude of the scattering rate used for the polariton-polariton interaction versus polariton density is shown in Figure 5.5. At low density for CW and for

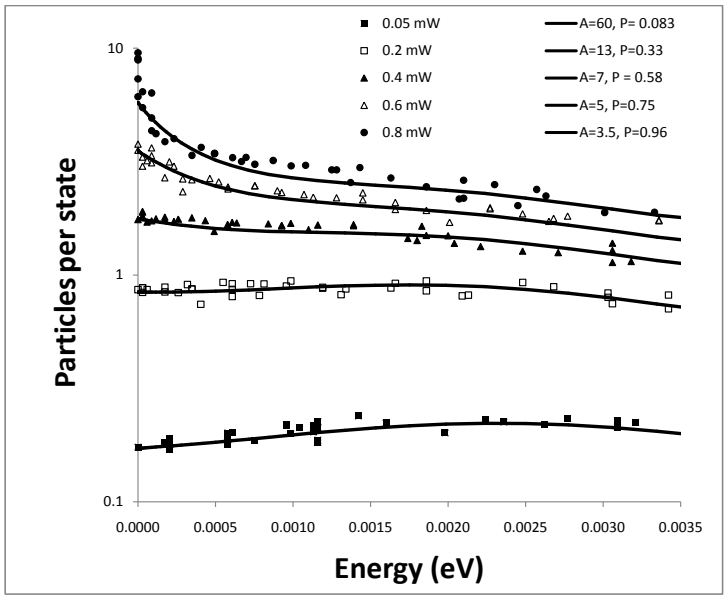

Figure 5.4: A fit to the quasi-CW pumped data using polariton-polariton and polariton-phonon scattering. "A" stands for the coefficient used in front of the polariton-polariton scattering cross-section and "P" is the generation rate used. Simulated plots are shown next to their corresponding experimental pump power.

quasi-CW the necessary coefficient to the scattering rate is inversely proportional to the density of the polaritons. The need for this can be seen in the data of Figures 5.3 and 5.4 because the polariton steady-state distribution stays essentially the same as density approaches zero. One would expect the polariton-polariton cross section to be constant. This would imply polariton-polariton scattering becomes unimportant at low density and

only polariton-phonon scattering is important. However, polariton-phonon scattering alone would not give the low density distributions. We thus needed another scattering mechanism that would be constant at low density.

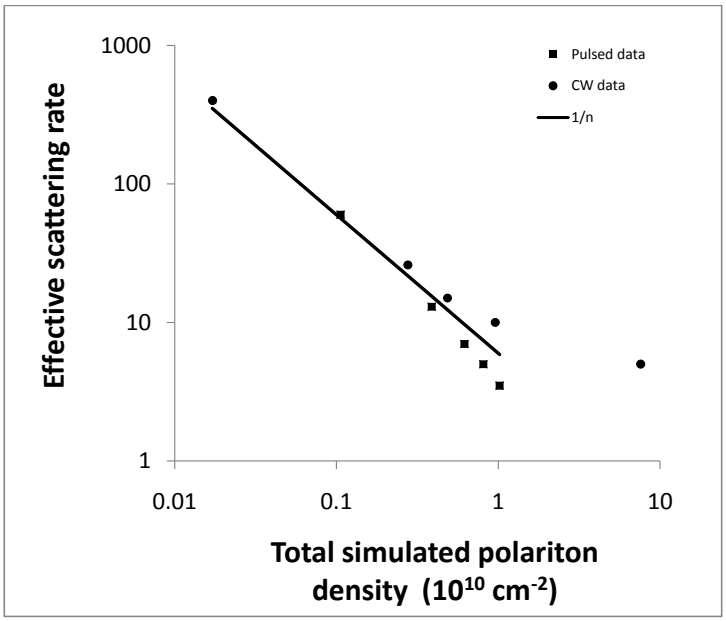

Figure 5.5: Plot of the coefficient used for the polariton-polariton scattering matrix element as a function of simulated polariton density.

5.2.2 FITS USING POLARITON-ELECTRON SCATTERING AND PIEZOELECTRIC SCATTERING

As discussed above, we reasoned that there was some scattering process that was constant as polariton density decreased. The flatness of the polariton distribution at low density, as

shown in Figure 4.14 and Figure 4.15 led us to investigate another process. The deformation potential interaction with the polaritons is relatively weak compared to the polariton-polariton and polariton-electron interactions and has little effect on the lower polariton distribution, hence the bottleneck. However, the piezo-electric phonon interaction is inversely related to the momentum exchanged. We considered that for the small momentums of the lower polaritons they might be strongly influenced by this effect. Our estimates concluded that the piezo-electric interaction was stronger than the deformation potential interaction for small exchanges. When considering all possible interactions, however, by using the simulation, we found the piezoelectric scattering to be only a small term. The results were essentially the same with or without the piezo-electric interaction as parameters were varied. This is because most of the piezo-electric interaction happens within the k-space defined by the width of each bin. Nevertheless, we kept the scattering mechanism in the total model. This is primarily because the phonon process is easily calculated and only increases the time required to process the simulation by 2%.

Next, we hypothesized that a small population of free electrons would create a constant scattering process as polariton density was decreased. When polariton-electron scattering was included, we found we could get good fits at high density when the polariton-polariton cross-section was increased only about 20% greater than the literature value (see Equation 2.20). We then found the cross-sectional coefficient for the polariton-electron interaction that would match the lower-density polariton data with a nearly constant density of electrons, which we estimated to be around $\sim 2 \times 10^8$ cm^{-2}.

Using a constant value for the polariton-polariton cross-section that is 20% larger than the literature value, we found that we could fit all the data if we allowed the lattice temperature and free-electron density to vary over reasonable ranges. Figure 5.6 shows how the chosen simulated generation rate correlates to the experimental pump intensity for the CW data and Figure 5.7 is for the quasi-CW data. Essentially, the fits show a linear dependence. The simulated polariton and electron densities as a function of the simulated generation rate are shown in Figure 5.8 and 5.9 for CW and quasi-CW. The polariton density begins to saturate with increasing pump power. This is to be expected. As the density increases the polaritons get shifted to lower momentum values. These states have shorter lifetimes than

the higher energy states. Thus it takes a higher generation rate to keep a steady state. The electron density is essentially constant.

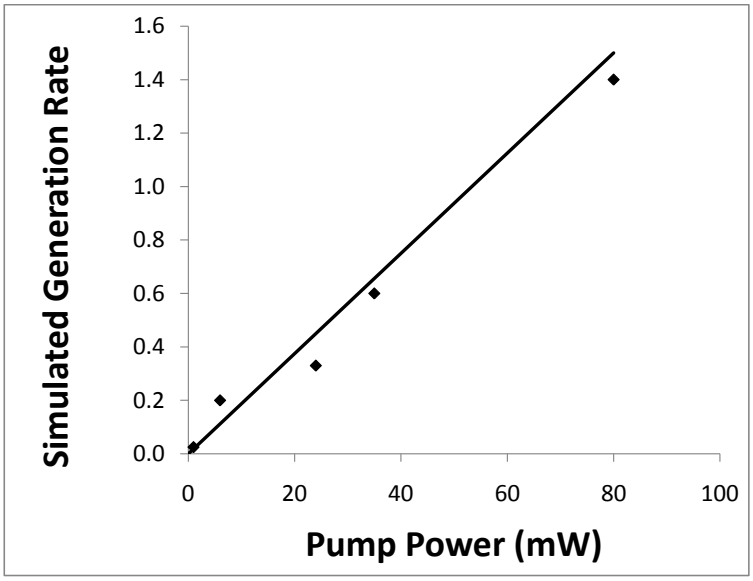

Figure 5.6: Plot of the simulated generation rates to the corresponding experimental pump powers. The line is a guide for the eye.

The final best fits to the CW data are shown in Figure 5.10. Keeping the polariton-polariton scattering cross-section the same as was used for the CW fits, we made fits to the quasi-CW data to check for consistency (see Figure 5.11). As in the case without the polariton-electron interaction, the fit to the CW data again has a larger spread in temperatures compared to the fits for the quasi-CW data.

The fits to the data are good, and the parameter values are physically reasonable. We found reasonable fits when the polariton-electron scattering cross section was a constant

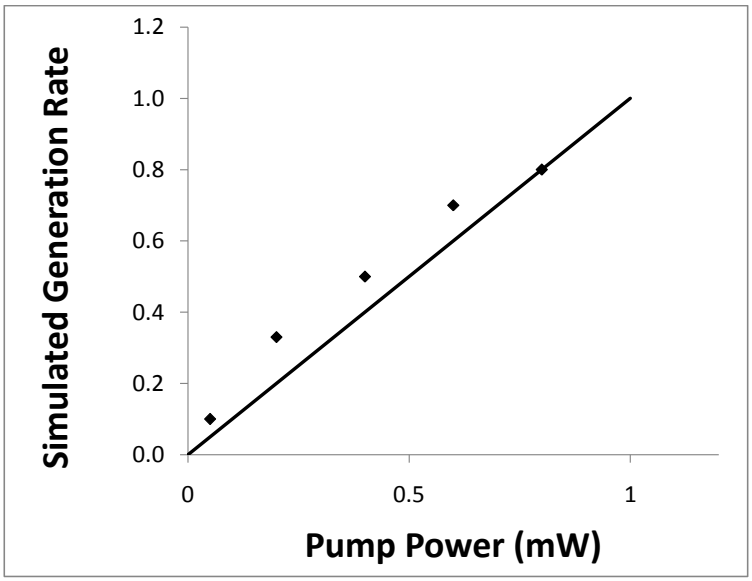

Figure 5.7: Plot of the simulated generation rates to the corresponding experimental pump powers. The line is a guide for the eye.

factor of 30 times larger than the value for the theory presented in Chapter 2 (see Equation 2.56) and for simulated electron densities around 7 x 10^8 cm^{-2}. There is a large amount of uncertainty in the cross-section for electron-polariton scattering [35]. Experimentally, we don't have a measured electron density. Theoretically, there is also much uncertainty in the value of the electron-polariton scattering cross-section. As such, we do not claim these fits to be a measure of the electron-polariton scattering cross-section.

Where do these electrons come from? Figure 5.9, shows that the electron densities are

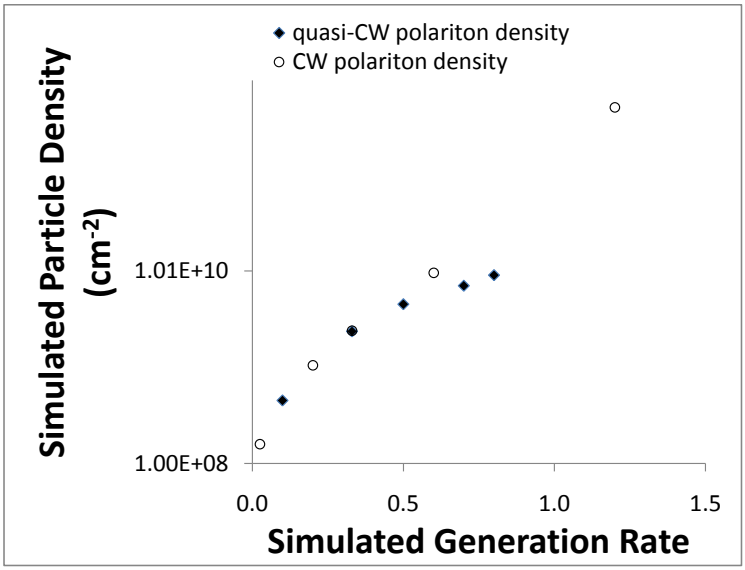

Figure 5.8: Plot of the steady state simulated polariton density as a function of simulated generated rate.

nearly constant with an average density of 7 x 10^8 cm^{-2}. One explanation for this is that there is an induced electron density at the interface between the GaAs and GaAlAs making the quantum wells [54]. The stress used in the experiments creates a piezoelectric polarization in each material. At the interface there is a polarization mismatch. This mismatch manifests itself as a surface charge. We provide the following estimate for this surface charge.

The effect of strain in the system is based on the Pikus-Bir strain Hamiltonininan [31],

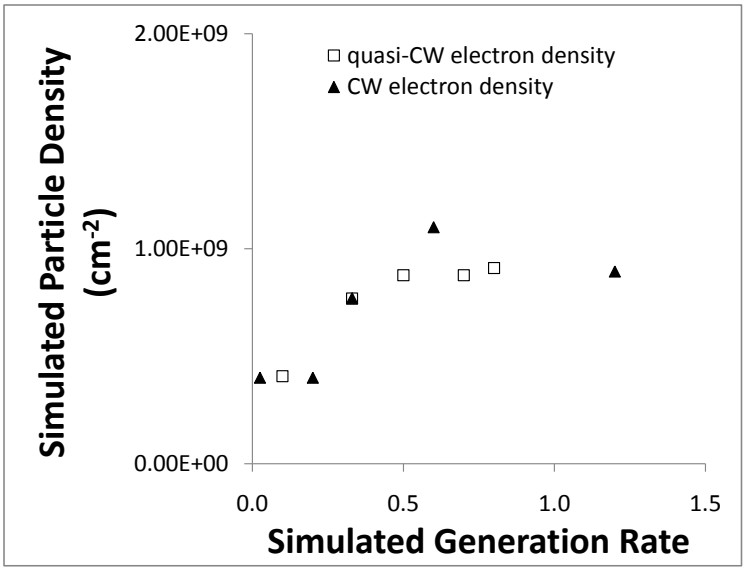

Figure 5.9: Plot of the total simulated free electron density as a function of simulated generated rate.

$$H_{PB} = a(\epsilon_{xx}+\epsilon_{yy}+\epsilon_{zz}) + 3b\left[\left(J_x^2 - \frac{j^2}{3}\right)\epsilon_{xx} + c.p.\right] + \frac{6d}{\sqrt{3}}\left[\frac{1}{2}\left(J_xJ_y + J_yJ_x\right)\epsilon_{xy} + c.p.\right], \quad (5.4)$$

where a is the hydrostatic deformation potential, b and d are shear deformation potentials, ϵ_{ij} is a strain component, and $J's$ are the spin states of the valence band, $m =$

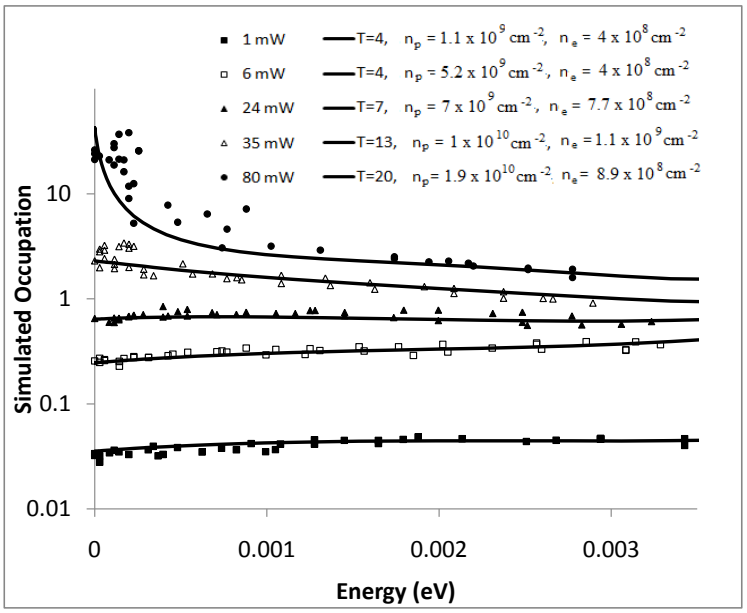

Figure 5.10: The final fits to the CW experimental data. In the legend, "T" stands for the simulated lattice temperature, "np" is the simulated polariton density, "ne" is the simulated electron density. Simulated plots are shown next to their corresponding experimental pump power.

$3/2, 1/2, -1/2, -3/2$. The deformation potentials are the same that were used in the simulation. For a band shift of about 15 meV, as in our experiments, then the strain is on the order 10^{-4}. The polarization, P, induced by piezoelectricity in a material is given by,

$$P_i = e_{ij}\epsilon_j, \tag{5.5}$$

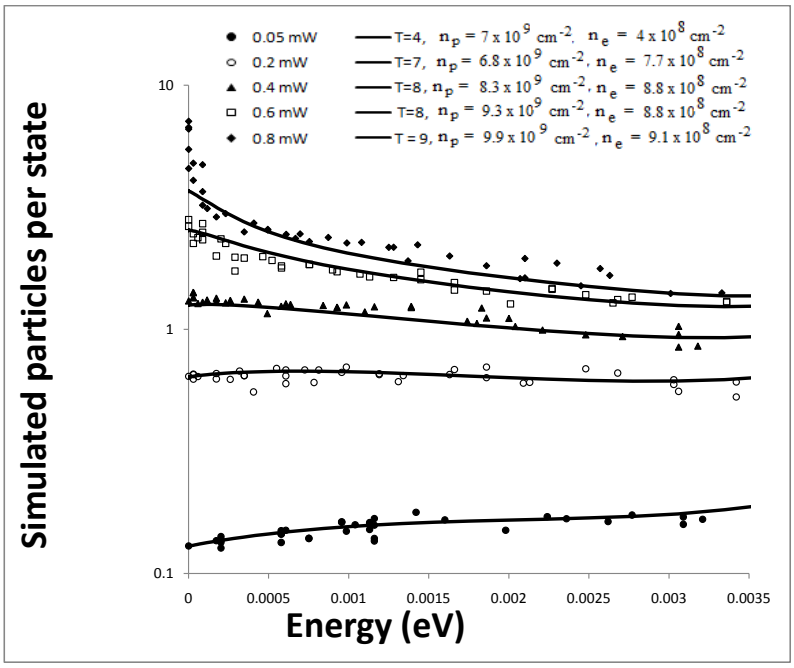

Figure 5.11: The final fits to the quasi-CW experimental data. In the legend, "T" stands for the simulated lattice temperature, "np" is the simulated polariton density, "ne" is the simulated electron density. Simulated plots are shown next to their corresponding experimental pump power.

where e_{ij} is the piezoelectric constant. For $Ga_{0.8}Al_0.2As$, $e_{41} = 0.173$ C/m^2,[55] and for GaAs, $e_{41} = 0.16$ C/m^2[33]. The difference in the two polarizations provides a surface charge at their interface. The estimate is an electron density of 8 x 10^8 e/cm^2, close to the average value of electron density used in the simulations. Again, though, there is much uncertainty. We have twelve quantum wells in the sample and there are two GaAs-GaAlAs interfaces for

each quantum well. This is a possible explanation why the scattering cross-section needs to be much larger than the expected value. Another possibility is that there is the presence of low density donor impurities. These may contribute electrons if they become ionized by local electric fields caused by the stress. Instead of there being free electrons, another possibility is that there is disorder in the quantum well widths. Polariton-structural-disorder interactions have been discussed as inducing coherent elastic scattering[56][57]. This disorder introduces a broadening of the polariton states and we have not ruled out this as a posible explanation of at least some of the broadening that we see in the experimental data. The broadening of the states means that the delta function used for the density of states in Fermi's Golden Rule becomes a Lorentzian,

$$\delta(E) \rightarrow \rho \propto \frac{\Delta E}{(E - E_o)^2 + \Delta E^2}. \tag{5.6}$$

Here, ΔE is the broadening caused by the disorder, E_o is the nonbroadened energy, and ρ is the density of states. At this time we have not thoroughly studied this possibility and the effect it might have on the distribution curves.

We give a final plot for this chapter, Figure 5.12. This plot shows three different numerical simulations for the highest pumping density of the quasi-CW data. The first case is for only polariton-phonon interaction, the next uses the full simulation (polariton-phonon scattering, free electron-polariton scattering and polariton-polariton scattering) but no Bose statistics, and the third includes Bose statistics. We conclude that the upturn in the polariton distribution is really the effect of Bose statistics.

Recent theoretical works have shown that lack of complete thermalization does not prevent the polaritons from making a phase transition[58, 59, 60]. Instead, a bimodal distribution occurs. The higher energy polaritons, being constantly heated from higher energy particles form one distribution. The lowest polariton energies, where the upturn is, form another distribution and can make a phase transition to a condensate. Simulations, like ours, cannot model the onset of coherence since they use Fermi's Golden Rule, which assumes incoherence. These simulations can show the buildup of population due to statistics just before coherence appears.

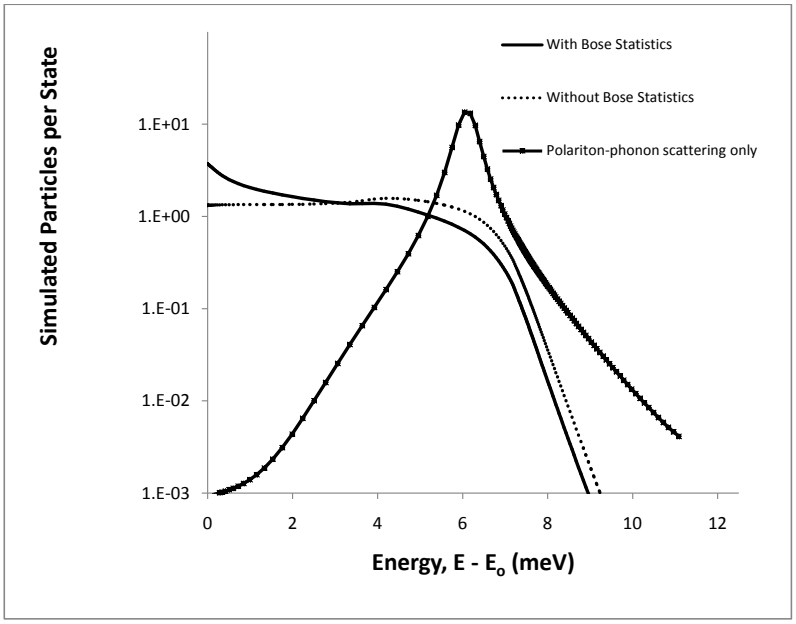

Figure 5.12: The steady state results of using (1) polariton-phonon interactions, (2) polaritons interacting with polaritons, electrons and phonons, but no Bose statisitcs, and (3) polaritons interacting with polaritons, electrons, and phonons with Bose statistics included.

6.0 FUTURE DIRECTIONS AND CONCLUSION

6.1 ACCOMPLISHMENTS

We set out to show that polariton microcavities are good candidates for achieving a solid-state BEC. Experimentally, we have taken data that is highly suggestive of the existence of a BEC. However, the distribution functions are definitely not completely thermalized and we wanted an explanation for their appearance. The numerical model has provided a good fit to data and confirmed that the accumulation at $k = 0$ at the highest densities presented is due to Bose statistics. Also, the numerical model has shown that a possible explanation for the flat distribution curves at low density is that the polaritons are interacting with a population of electrons. At the time of writing this thesis we did not have experimental evidence that there are free electrons in the system and do not have a strong case as to why there should be any free electrons. Using the numerical model, we intend to investigate other interactions that may explain the distribution curves through the possibility that there is disorder in the quantum well structure. This may be a cause of the energy broadening in the polariton states and lead to a breakdown of energy-momentum conservation when Fermi's Golden Rule is applied.

The models have also shown that the increase in population of the low energy polaritons are due to bosonic interactions. This is the beginning of the onset of a macroscopic occupation of a single state, a necessary step toward reaching coherence. The model presented here cannot model coherence, since it assumes Fermi's golden rule, which implies incoherent processes. Nevertheless, we can model right up to the onset and show that the peaking is truly due to bosonic effects.

This thesis has provided a model that accurately describes the microcavity polariton

dynamics. It is capable of describing how to get to the experimental steady-state results. Showing how to obtain the results of an experiment through calculation is one use of a model. Another use is how the model can be extended and used to suggest further experiments. The polaritonic system is very complicated with many interactions taking place. The model allows for parameters to be varied, detuning for instance, to see if the dynamical rates can be increased and produce steady state results which are more favorable for coherent effects.

6.2 WHAT'S NEXT

The conclusion of one set of experiments always leads to another set of experiments. There are numerous paths to be taken. I suggest a few here.

The experimental data shows blueshifts in the energy dispersion curves. The models have the ability to calculate the magnitudes of these shifts. An exploration of how the interactions affect the polariton dynamics may provide important information as regards to experimental directions and microcavity engineering.

Experimentally, additional time resolved spectroscopy measurements could be a direction to go. Experiments have shown that it is difficult to determine the polariton lifetime. Measurement of the uncoupled excitonic lifetimes and photonic lifetimes independently may resolve this problem. The experiment would have to be done in a region of the sample where the two modes are uncoulpled. Knowing the two modes separately would allow the polariton lifetime to be calculated.

What would we expect the results to be without the strain induced charge? Figure 6.1 shows an example of what the model provides without the polariton-electron scattering included. The simulation suggests that the bottleneck would persist under these pumping conditions.

This thesis has suggested that the presence of free electrons is primarily induced by stress used in the trapping method. Figure 4.16 shows that without stress the polaritons are stopped by the bottleneck. As stress is increased, the polaritons flow down to lower energy states. Perhaps this is because the system is far from resonance in the upper picture

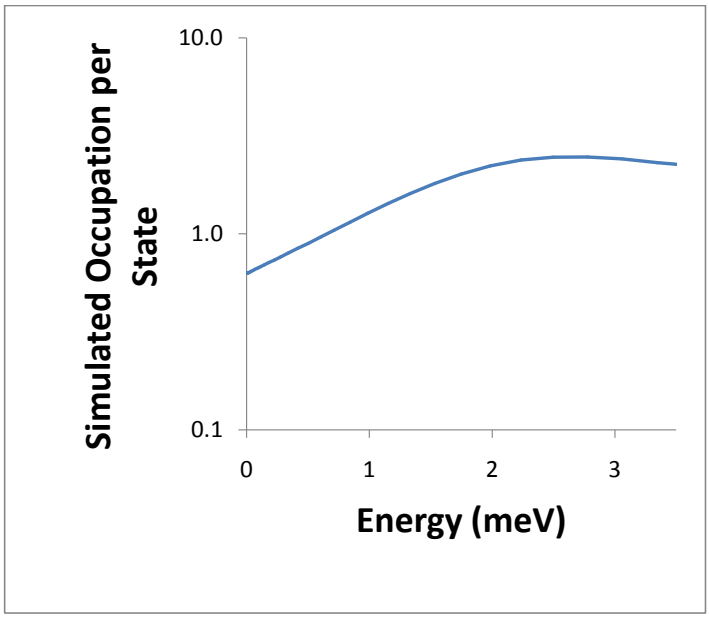

Figure 6.1: The steady state simulated distribution function for the polaritons using the same parameters as the highest generation rate for the quasi-CW data without the free electron-polariton interaction included.

of Figure 4.16 or perhaps it is because there are no free electrons. A next step would be to take the data without stress but in resonance. If the polaritons were still kept above the bottleneck, this would further the assertion that the polaritons make it past the bottleneck because of the stress induced charge.

APPENDIX A

KINETICS OF BOSON-BOSON SCATTERING IN A 2D FLAT POTENTIAL

Numerous analytical descriptions of phase transitions for composite bosons in various two-dimensional systems have been reported. In this section we offer numerical support to those descriptions which apply to a superfluid transition for free bosons. We continue to only consider the bosonic nature of the composite particles and completely disregard any fermionic composition. The bosonic nature studied is stimulated scattering and the allowance for states to have an occupation greater than one.

As discussed previously, it is analytically proven that free bosons only form a BEC when the dimensionality of the system is greater than two. However, a phase transition to a superfluid state may still occur. Kosterlitz and Thouless[10] presented a theoretical description for this transition. Their work puts an upper limit on the required density of the superfluid state at transition as that of the two dimensional quantum concentration, n_Q. We use n_Q as our unit of density and assume the superfluid density, $n_S = n_Q$, at transition. Along with a superfluid component there also exists a normal component, n_N, and together they make up a total density,

$$n_T = n_S + n_N. \qquad (A.1)$$

Kosterlitz and Thouless put the density of the normal component of helium as it undergoes a transition around $3.5n_Q$. The normal component density is studied in this paper.

Berman, *et al*[61] presented results for a model of indirect excitons in double quantum wells with and without an external random potential. Their part of the theory without an

external potential applies to the results of this work. That theory shows the density of the normal component after a superfluid transition occurs is $n_N = 0.4 n_Q$.

Fischer and Hohenberg[62] presented results based on Bogoliubov theory for a weakly interacting Bose gas in two dimensions. In a sufficiently dilute system the normal component density is on the order of the quantum concentration when a transition occurs.

The theory for our calculations follows that presented in Chapter 2 for polariton-polariton interactions except that we exclude the fractional part due to the polariton. We continue to use the matrix element as a constant. This models hard core scattering.

A.0.1 Results

One possible initial condition is to assume that bosons are injected into the system with a gaussian distribution. For low density the average energy is $k_B T$. As the density increases this average energy must decrease to keep the same temperature. Figure A1 shows a plot of a low density, $n < 0.01 n_Q$, distribution. By the fifth scattering event the profile of the initial distribution has disappeared and the distribution begins to look like a Boltzmann distribution. A least squares fit is made to this distribution using the chemical potential and temperature as parameters. Figures A2 shows how these parameters evolve in time, normalized to their equilibrium values for three different densities all starting at the same temperature. For low densities, $n < 0.01 n_Q$, the distributions evolve at the same rate and are very near equilibrium after about twenty scattering events. For $n = n_Q$ the systems reach their equilibrium values after about forty scattering events. Note, however, that the actual time for this density to come to equilibrium is much less than the lower density. While it takes twice as many scattering events each scattering event is rouchly 75 times shorter based on classical hard core scattering calculations. When $n = 3.6 n_Q$ the system is still far from equilibrium after sixty scattering events, $T = 1.55 T_L$ and $\mu = 4.2 \mu_0$. Note that this effective T is lower than the critical temperature for a system with this density.

The results presented are based on using an equilibrium distribution and calculating the scattering per particle in the lowest energy bin. At a constant temperature the chemical potential can be assigned to give a specific distribution based on the Bose-Einstein distribu-

tion,

$$f(E_i) = \frac{1}{exp[\beta(E_i - \mu)] - 1} \qquad (A.2)$$

This leads to a particular density. The density of the normal component for such a distribution is then found from

$$n_N = \int D(E) f(E) dE \qquad (A.3)$$

where in the flat potential two dimensional system the density of states per unit area, $D(E)$, is a constant. To make our plots the temperature is held fixed and the chemical potential is changed over an array of values. The equilibrium distribution function and density are then calclulated.

Using equilibrium solutions, the scattering rate per particle in the lowest energy bin is calculated as a function of density at a constant temperature. Figure A3 shows a plot of this scattering rate for densities below the quantum concentration. At low densities a linear dependence on density is expected. A line is fit to the first 50 points. The plot begins to deviate from this line around $n_N = 0.3n_Q$. For $f(0) = 1$ the scattering rate is twice as large as that for the linear fit. This occurs for $n_N = 0.67n_Q$. Figure A4 shows the same data over a larger range of densities. This log-linear plot shows an exponential dependence as shown by the straight line. The solid line represents an exponential fit to the results at the highest densities. Figure A5 shows a close up of Figure A4. It is seen that the change in scattering behavior asymptotically reaches a constant exponential dependence around $n_N = 3n_Q$.

A.0.2 Conclusion

We assume that a change in behavior of the scattering rate is indicative of when a phase transition takes place. The numerical results of this model imply some ambiguity as to when the transition to a superfluid occurs. An argument could be made for $0.3n_Q < n_N < 3n_Q$. These values are consistent with what can be inferred from a broad range of other models. Indirect excitons in double quantum wells without a random potential are near the low end of this scale. Weakly interacting Bosons are within this range. Films of helium are at the upper end.

Comparison shows that the 2D system equilibrates at about the same rate as the 3D system. Low density systems will be near equilibrium within twenty scattering events. Densities at the quantum concentration will take around forty scattering events. Densities above the quantum concentration will take many more scattering events.

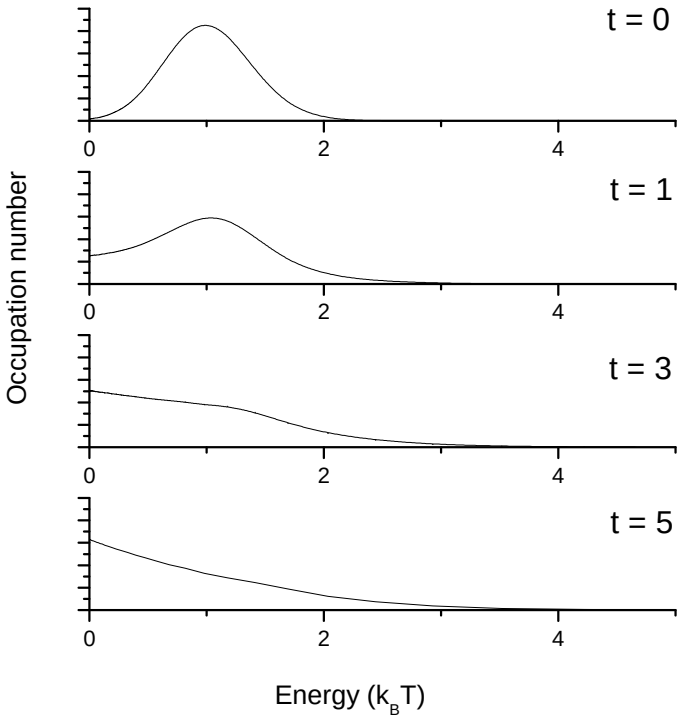

Figure A1: Low density energy distribution at scattering times 0, 1, 3, 5

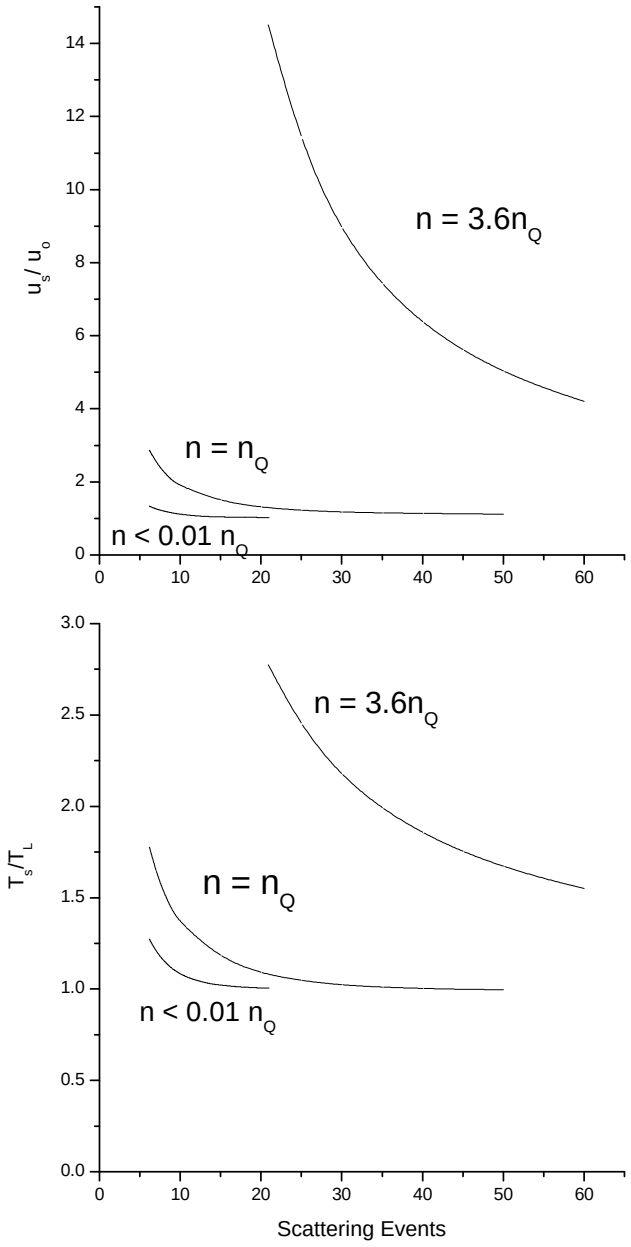

Figure A2: The evolution of the fitting parameters, chemical potential and temperature, to the distributions for three different densities. Time is measured in scattering events. The chemical potential is in units of the equilibrium chemical potential and the temperature is in units of the lattice temperature.

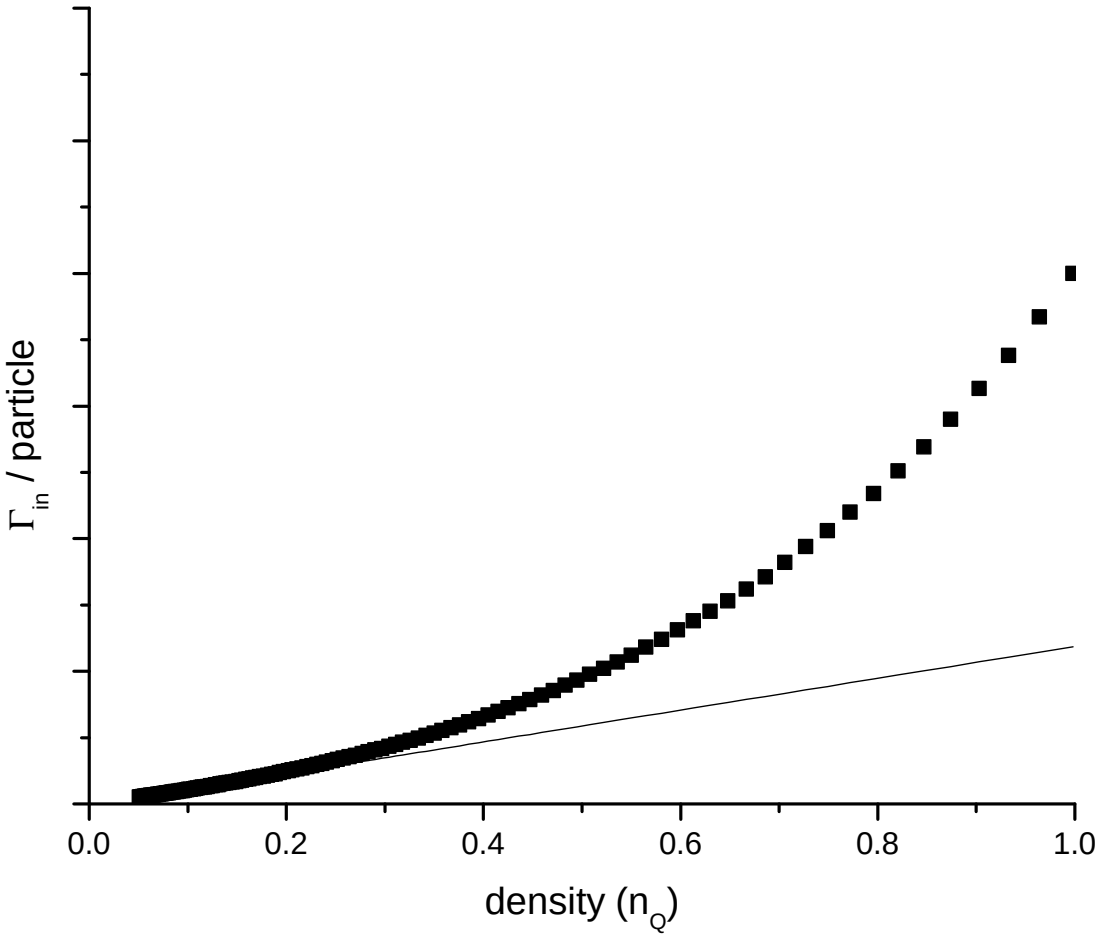

Figure A3: Scattering rate per particle in the lowest energy bin as as a function of density below the quantum concentration

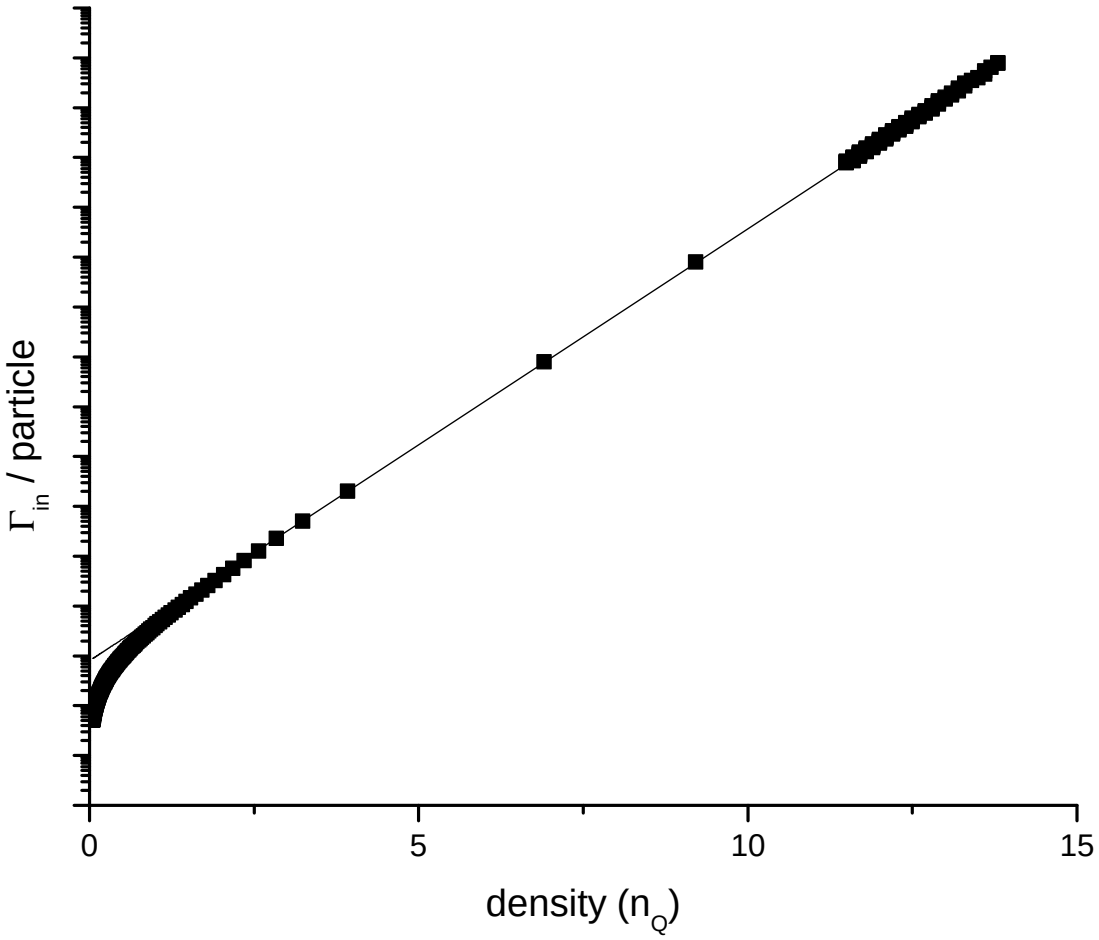

Figure A4: Scattering rate per particle in the lowest energy bin as as a function of density

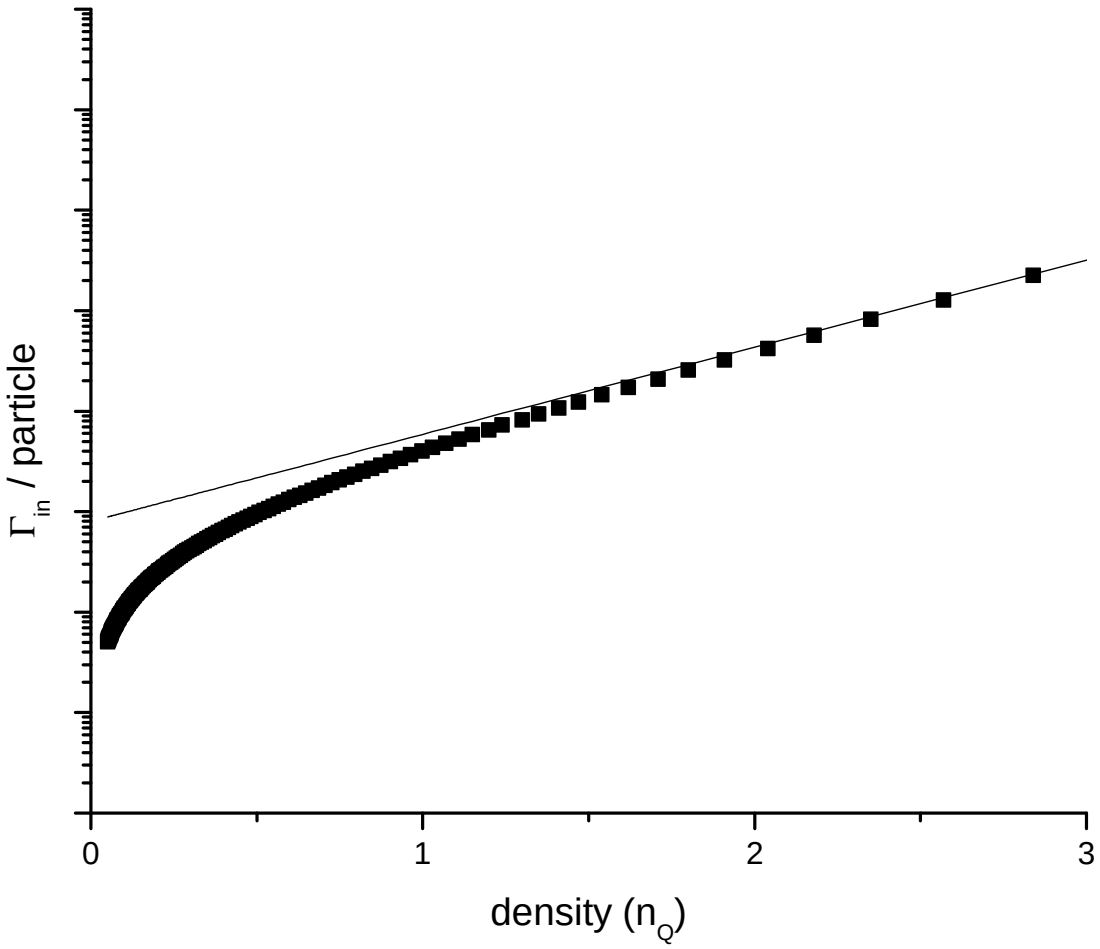

Figure A5: Close up of figure 4.

APPENDIX B

FULL CODE

The code used in this thesis is presented here. The code uses SI units except for energies which are in electron-Volts. The main C program is Integrate.c. It is listed first, here. All header files are listed afterwards in the order that they appear in the #include list.

B.1 INTEGRATE.C

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include "constants.h"        /* file where constants are defined */
#include "variables.h"        /* file where variable are defined */
#include "parameters.h"       /* file where important parameters for a specific calculation are defined */
#include<stdlib.h>
#include<string.h>
/*#include "Errors.h" */
#include "polpara.h"
#include "Ecset.h"
#include "getk3.h"
#include "dEdk.h"
#include "DOS.h"              /* function for calculating the density of states */
#include "rand.h"
#include "del.h"              /* function for creating the mesh for the calculation */
#include "getf4.h"
#include "getb4.h"
#include "polcheck.h"
#include "polEnergy.h"
#include "polfraction.h"      /* function for determining the excitonic fraction of a polariton */
#include "ktilii.h"
#include "gaussian.h"         /* gaussian f vs E */
#include "boltzmann.h"        /* boltzmann distribution */
/*#include "boltzmann2.h" */
#include "pulseadd.h"         /* pulse input, thermalized */
#include "pulsegauss.h"       /* pulse input, gaussian */
#include "pulseflat.h"        /* pulse input, all states equally populated */
/*#include "pulseflat2.h" */
#include "neareq.h"           /* input an instantenous nearly thermalized distribution */
#include "phononfvsE.h"
#include "updatef.h"          /* function for updating  occupation numbers for scattering, decay, and pumping*/
#include "polpMatx.h"         /* determine the scattering matrix elements for polariton-longitudinal acoustic phonon scattering */
#include "polelMatx.h"
#include "polpTAMatx.h"       /* determines the scattering matrix elements for polariton-transverse acoustic phonon scattering */
#include "polpTA_PiezoMatx.h"
#include "polpFmatx.h"        /* determine the scattering matrix elements for polariton-optical phonon scattering */
```

```
#include "bospMatx.h"         /* determine the scattering matric elements for exciton-exciton scattering */
#include "findf.h"
#include "3Dbosescat.h"       /* 3D boson-boson scattering integral */
#include "3Dfermiscat.h"      /* 3D fermi-fermi scattering integral */
#include "2Dbosescat.h"       /* 2D boson-boson scattering integral */
#include "2Dpolscat.h"        /* 2D polariton-polariton scattering integral */
#include "2Dpolelscat.h"
#include "2DpolFscat.h"       /* 2D polariton-optical phonon scattering integral */
#include "2DxpEexchange.h"    /* 2D exciton-acoutic phonon scattering integral */
#include "2DpolpEexchange.h"
#include "3DxpEexchange.h"
#include "2DpolpLAscat.h"     /* 2D polariton-3D longitudinal acoustical phonon scattering integral */
#include "2DpolpTAscat.h"     /* 2D polariton-3D transverse acoustical phonon scattering integral */
#include "2Dbospscat.h"       /* 2D exciton- 3D acoustical phonon scattering integral */
#include "Vvsq.h"
#include "uniform.h"          /* uniform f vs E  */
/*#include "datafit.h" */
#include "fvsEsave.h"         /* function for saving calculated values */
#include "GQtest.h"
#include "gauleg.h"           /* function for generating the Gaussian Quadrature points and weights */
#include "fromfile.h"         /* function for reading in values from a text file */
#include "initiate.h"         /* function for directing the initialization of the calculation */
#include "scat.h"             /* function for directing the type of scattering to occur */
/*#include "MinEn.h" */
#include "polReNorm.h"        /* function for renormalizing the dispersion curve of polaritons */
#include "zeta.h"

int main()
{
inj1 = inj;
Ntau = 0;
indexr = sqrt(einf);
kcz = setkcz(detune, Exo, kxy); /* in polpara.h */
Ec = setEc(kcz, kxy);
gauleg(-1,1,yy,ww,GQp);
mu *= kb * TT;

/*if (initial == 1)
{*/
Tm = zeta(2, exp(mu / (kb * T))) / (-log(1 - exp(mu / (kb * T))));    /* zeta(dimensionality, ) */
/*
g = log(1 - exp(mu / (kb * T))) * log(1 - exp(mu / (kb * T))) / (2 * zeta(3, exp(mu / (kb * T))));
}*/

hw *= kb * T;
srand( (unsigned)time( NULL ) );         /* seed the random numbers from clock*/
m = -1;                                  /* m = # of steps to max energy for uniformfvsE */
                                         /* determined in uniform.h */
initiate(initial, del, p, DOS, m, f, Nin, Nout, N, mu, hw, expp, Tm, statype, Ec, indexr);   /* initial particle distribution */
if(initial == 0) initial = 9; /* after reading in the data a flat pulse is added in */
Ntotal = 0;
Etotal = 0;

/*  determine total number of particles and total energy */
for(i = 0; i <= p; i++)
{
     Ntotal += N[i];
     Etotal += N[i] * (f[i][0] + f[i+1][0]) / 2;
}
Ntotal += N[p+3];
Nbegin = Ntotal;

fvsEsave(del, 0, N, f, Nin, Nout, tau, taut, dataname, Etotal, Norf, fname, y, o, Ntotal, Geo, kpp, Ninpol, Ninpolel, Ninph, Avetau);

do
{
/* do the scattering integrals */
scatter(scattype, f, Nin, Nout, m, zzz, del, Geo, kpp, delk, y, indexr, kcz, disp);
/* update the occupation numbers for scattering, pumping, and recombination */
x = updatef(Ntotal, uprate, x, dN, N, f, Nin, Nout, Geo, kpp, Stime, taut, Avetau);
/* renormalize polariton distribution if set for in parameters.h */
if(disp == 3)
    ReNorm(indexr);
/* determine total number of particles and total energy */
Etotal = 0;
Ntotal = 0;
dNtotal = 0;
for(l = 0; l <= p; l++)
{
     Ntotal += N[l];
```

```
     dNtotal += Nin[l] * del[l];
     Etotal += N[l] * (f[l][0] + f[l+1][0]) / 2;
}
Ntotal += N[p+3];
/* determine time step */
change = x / Ntotal;
tau += change;
taut += Stime[p+3];
Ntau += Stime[p+3] * N[p+3];
if(Stime[p+3] < 1e-16) uprate[0] *= 1.1;

if(uprate[0] > 1) uprate[0] = 1;
/* record the calculation at the interval set in parameters.h */
if(y > (n1 * count) - 1)
{
     fvsEsave(del, n1, N, f, Nin, Nout, tau, taut, dataname, Etotal, Norf, fname, y, o, Ntotal, Geo, kpp, Ninpol, Ninpolel, Ninph, Avetau);
     n1++;
     x = 0;
}
y++;


sprintf(outfile3, ynum);
ft = fopen(outfile3, "w");
fprintf(ft, "y = %d\n", y);
fclose(ft);
}
while((n1*count <= o)  && (taut < maxtime));

}
```

```
double D2A = 0.0001;            /* 2D sample area in m^2 */
double aB = 130E-10;             /* exciton Bohr radius in m */
double beta = 2;                /* polariton - exciton volumic oscillator strength */
const int dg = 4;               /*  degeneracy  */
const double dn = 1.28E20;      /* particle density in particles/m^3  */
const double e = 1.2;           /*2.718281828;*/ /* in energy step size. see del.h */
double EB = 0.010;              /* exciton binding energy in eV */
const double eC = 1.60219E-19;      /* charge on an electron in Coulombs */
#define Ex0 1.61945             /* polariton --> exciton base energy in eV */
double einf = 10.9;             /* dielectric constant at infinity, set for GaAs */
double estat = 12.5 ;           /* static dielectric constant */
#define em 9.1095E-31              /* electron rest mass */
#define eo 8.85E-12             /* permitivity of free space, C^2 / N m^2 */
double Exo = 1.61945;
const double gs = 0.0149;       /* polariton splitting at resonance in eV, 2 times the Rabi frequency */
#define hb 6.5822E-16           /*  planck's constant  in eV s */
#define Itaup 0.2e12 /* 0.2e12 */         /* inverse of photon lifetime, s^-1 */
#define Itaux 1e6 /* 1e6 */              /* inverse of exciton lifetime, s^-1 */
#define kb 0.00008617           /* boltzmann constant in eV/K */
double Lata = 5.6533E-10;       /* Lattice constant, set for GaAs, meters */
double Lz = 7E-9;               /* quantum well thickness, meters */
double Me = 0.067;              /* exciton electron mass, in em, electron rest mass Piermarocchhi PRB 53 15834 (1996)*/
double Mh = 0.18;               /* For exciton total mass, exciton hole mass, in units of em, electron rest mass */
double Mh1 = 0.08;              /* Used for calculating Beta in polariton-photon interaction */
const double Pd = 6;            /* 3D crystal density */
double piezo14 = 0.16;          /* piezoelectic constant, set for GaAs e_(14) */
double qo[1];                   /* screening parameter coefficient, e^2 / (2 * e(inf)) */
#define pi 3.14159265358979
const double Sd = 5316;         /* set for GaAs */       /* 2D density kg/m^3 */
double sig = 0.5; /*0.00001; 0.000196764; */    /* injected gaussian width */
const double T = 10;             /* Lattice temp in K */
#define vc 2.998E8              /* speed of light, m/s */
#define v 5.117E3               /*rough average for GaAs */ /* longitudinal speed of sound in medium, m/s  */
#define vTA 3.012E3             /* transverse speed of sound in medium, m/s */
#define  wLO 1.070E13           /* longitudinal optical phonon frequency, GaAs */

/* see parameters.h for count, g, o, p, uprate, and others */
```

B.3 PARAMETERS.H

```
int count = 1000;              /* number of iterations per data writing */
double defpote = -7;           /* hydrostatic deformation potential in eV for electron, (Pikus-Bir, "a" for conduction band) */
double defpoth = 2.7;         /* hydrostatic deformation potential in eV for hole, (Pikus-Bir, "a" for valence band) */
double defpoteT = 0;          /* shear deformation potential in eV for electron, (Pikus-Bir, "b" and "d" for conduction band */
double defpothT = 3.8;            /* shear deformation potential in eV for hole, (Pikus-Bir, ((4/5)*(b^2 + d^2/2))^1/2, GaAs --> b = 1.8, d = 5.4 */
int delc = 5;                 /* key for del.h
                                 0: not sure
                                 1: e ^ i, e is in constants.h
                                 2: i ^ 2
                                 3: i ^ 3
                                 4: uniform
                                 5: polariton i
                               */
int disp = 2;                 /* dispersion relationship to be used,
                                 1 = exciton
                                 2 = polariton
                                 3 = renormalized
                               */
double detune = 0.00;         /* polaritons --> cavity mode detuning from resonance with excitons, eV */
int dopiezo = 2;              /* use piezoelectric interaction with phonons
                                 1 = no
                                 2 = yes
                              */
double g = 0.1;               /* initial occupation number for uniform dist., not used right now */
const int Geo = 5;            /* free particle dimensionality (1,2,3), harmonic potential (4), free polariton (5) */
const int GQp = 25;           /* Number of GQ points */
double hw = 0.001;            /*0.001 */   /* harmonic potential ground state in kb T */
#define inj 5e18        /*2E18  405.4725   1000  2000*/ /*injected gaussian density */
int initial = 9;           /* initial determines the initial configuration of
                              particles;
                              0: from file
                              1: uniform, fermi distribution
                              2: bose, equilibrium distribution
                              3: bose phonon distribution, gaussian free boson dis-
                                 tribution
                              4: gaussian distribution, free particles
                                 gaussian distribution, harmonic potential
                              5: 2D bose phonon distribution, gaussian free
                                 boson distribution
                              6: from near equilibrium
7: pulsed thermalized input
        8: pulsed gaussian input
                              9: pulse with flat occupation number distribution
                            */
const double kpb = 0.25;        /* base wavevector for polaritons  */
double kxy = 5330000;           /* polaritons --> injected in plane wavevector, m^-1 */
const double maxkp = 1.54E8;   /*1.54E8;  */    /* maximum k-parallel used in polaritons, m^-1 */
const double maxkT = 10;       /*10.5;*/  /* maximum energy used in calculation in kT */
double maxtime = 5e-1;        /* maximum time calculation will run */
double mu = -4;
double mue[1] = {-3.88};               /* chemical potential of free electrons in units of kT (eV) */
const char fname[20] = {"pol05.25.08e.d"}; /* data file name */
const char ynum[10] = {"y1value"};      /* file to check calculation progress */
const int Norf = 2;             /* flag for saving density(1) or occupancy(other) */
int o = 40000;                    /* number of iterations */
int p = 170;                    /* p + 2 is number of energy points */
double pulseT = 1e-8;           /* pulse length for intial = 7, 8, or 9*/
double Sa = 1;           /* confinement size of system for ground state, m^2 */
int scattype = 14;       /* type of scattering
                            1: 3D flat boson-boson
                            2: 2D flat boson-boson
                            3: 3D boson-phonon
                            4: 2D boson-phonon
                            5: gaussian quadrature testing
                            6: 2D polariton-polariton
                            7: stirctly 2D, polariton-phonon(edit before use, 7/24/04)
                            8: 2D polariton - 3D acoustical phonon(longitudinal and transverse)
                            9: 2D polariton - 2D polariton/3D acoustic phonon(longitudinal and transverse)
                           10: 3D flat fermi-fermi
                           11: 2D polariton - 2D polariton/3D acoustic and optical phonon
```

```
   12: 2D boson - 2D boson/3D acoustic phonon
                          13: 2D polariton - free electron / 3D acoustic phonon(longitudinal and transverse)
                          14: 2D polariton - 2D polariton / free electron / 3D acoustic phonon(longitudinal and transverse)
                          15: 2D polariton - free electron
                        */
int ScreenType = 2;      /* flag for the way the screening is handled,
                           1: epsilon -> epsilon * (1 + qo / (k - k'))
                           2: epsilon -> epsilon * (1 + qo * aB)
                        */
int statype = 1;         /* statistics used
                           1: Bose-Einstein
                           2: Boltzmann
                           3: Fermi-Dirac
                        */
double uprate[1] = {0.1}; /* {1E16}{0.4};*/  /* fraction of total number of particles
actually scattered per iteration */
double TT = 4;            /* Used for setting a different initial temperature than lattice */
double TTT[1] = {10};     /* Temperature of free electrons */
```

B.4 POLPARA.H

```
double setkcz(double det, double Ex, double kp)
{
return( (Ex + det) * indexr / (hb * vc) );
}
```

B.5 ECSET.H

```
double setEc(double k, double kx)
{
return(hb * vc * sqrt( (k * k) + (kx * kx) ) / indexr);
}
```

B.6 GETK3.H

```
double Determinek3(double E4, double f[1000][6], int zzz)
{
if (E4 == f[0][0])
    zzz = 0;
else if (E4 < f[p+1][0])
{
    while (E4 >= f[zzz][0])
    {
        zzz++;
        if (zzz > p + 1)
            zzz = 0;
    }
    zzz--;
    if (zzz < 0)
    {
        printf("error in getk3.h, zzz < 0\n");
        printf("E4 = %e, E(0) = %e\n", E4, f[0][0]);

        exit(1);
    }
    if(E4 < f[zzz][0] || E4 > f[zzz+1][0])
    {
        printf("k3 error\n");
        printf("E4 = %.20e, E(z) = %.20e, E(z-1) = %.20e, z = %d, point = %f\n", E4, f[zzz][0], f[zzz-1][0], zzz, point);
        exit(1);
    }
    point = (E4-f[zzz][0])/(f[zzz+1][0] - f[zzz][0]);
    ff4 = kpp[zzz] + (point * delk[zzz]);
}
else
    ff4 = kpp[p+1];
polD[0] = zzz;
return(ff4);
}
```

B.7 DEDK.H

```
double getdEdk(double EE, double KK, int disp, double indexr, int zzz, int y)
{

int up, down;
if(disp == 1)
{
EEKK = hb * hb * eC / (2 * em * (Me + Mh));
}
if(disp == 2 || (disp == 3 && y == 1))
{
      Exkt = Ex0 + (hb * hb * KK * KK * eC/ (2 * em * (Me + Mh)));
      Eckt = hb * vc * sqrt(kcz * kcz + KK * KK) / indexr;
/*    E2EE = (Exkt * Exkt) + (Eck * Eck) + (gs * gs) - (2 * EE * EE);
      E4EE = (Exkt / (Me + Mh)) + (vc * vc / (indexr * indexr));*/
/*    E3EE = 2 * Exkt * hb * hb * KK / (em * (Me + Mh));

dE/dk^2 ---> */
      EEKK = hb * hb * ((Exkt / (em * (Me + Mh))) * eC * (Eckt * Eckt - EE * EE) +



(vc * vc / (indexr * indexr)) * (Exkt * Exkt - EE * EE)) / (2 * EE * ((Exkt * Exkt) + (Eckt * Eckt) + (gs * gs) - (2 * EE * EE)));

/*
dE/dk -->    EEKK = ((2 * E3EE) / (4 * EE * E2EE)) * ((Eck * Eck) - (EE * EE));
*/
 /*     EEKK = hb * hb * KK * ( E4EE - (E2EE * ( (Exkt * Exkt + Eckt * Eckt + gs * gs) * E4EE - 2 * (Exkt * Exkt * Eckt / (Me + Mh) +
```

```
(vc * vc * Exkt * Exkt / (indexr * indexr))) ) / ( 2 * EE);*/
    if (EEKK < 1e-22)
        EEKK = 1e-22;
}
if(disp == 3 && y != 1)
{
    Determinek3(EE, f, zzz);
    up = polD[0] + 1;
    down = polD[0] - 1;
    EEKK = (f[up][0] - f[down][0]) / ((kpp[up] * kpp[up]) - (kpp[down] * kpp[down]));
}
return(EEKK);
}
```

B.8 DOS.H

```
/* number of states / unit volume / differential energy */
double DOSf(double DOS[1000], double f[1000][6], double indexr, int y)
{
DOS[p+3] = dg / Sa;     /* ground state density of states */
for (jj = 0; jj <= p + 1; jj++)
{
    if (Geo == 3)
        DOS[jj] = dg * em * Me * sqrt(em * Me * f[jj][0] / (2 * eC)) / (pi * pi * hb * hb * hb * eC);
    if (Geo == 2)
        DOS[jj] = dg * em * Me / (pi * hb * hb * eC);
    /*if (Geo == 1)
     {  DOS1 = 1 / sqrt(f[jj][0]);}*/
    if ( Geo == 4)
        DOS[jj] = (f[jj][0] / hw);
    if ( Geo == 5)
    /*  DOS[jj] = kpp[jj] / (2 * pi);  */
    /*    DOS[jj] = (dg / (4 * pi * hb * hb)) * f[jj][0] * (Elp1[jj] - (2 * f[jj][0] * f[jj][0])) /



(((Exk[jj] / (em * (Me + Mh)))  * ((Eck[jj] * Eck[jj]) - (f[jj][0] * f[jj][0]))) +



((vc * vc / (index * index)) * (Exk[jj] * Exk[jj] - f[jj][0] * f[jj][0]))); */
        DOS[jj] = (dg / (4 * pi)) / getdEdk(f[jj][0], kpp[jj], disp, indexr, 0, y);
}
return(1);
}
```

B.9 DEL.H

```
void delSet(double del[1000], double expp, int delc, double f[1000][6], double indexr)
{
int p1;  /* p - 20 */
switch(delc)
{
case 0:
/*  ----- uniform change in occupation number del ----- */
uu = log(1 - exp(-2*g));
fff0 = 1 / (exp(-uu) - 1);
fffm = 1 / (exp(maxkT - uu) - 1);
ch = 1 - exp(log(fffm/fff0)/p);
offs = log(1 / (1 + ch));
for (i = 0; i <= p+1; i++)
{
```

```
      del[i] = kb * T * ( offs + log((1 + fff0 + ch * fff0)/(1 + fff0)));
      fff0 = fff0 * (1 - ch);
}
break;

case 1:
/*  ----- exp(i) del ---- */
expp = 1;
for (j = 0; j <= p; j++)
       expp *= e;
/*printf("%e\n", expp);*/
dels = (expp - e) / ((e - 1) * maxkT);
expp = 1;
for (i = 0; i <= p + 1; i++)
{
       expp *= e;
       del[i] = kb * TT * expp / dels;
}
break;

case 2:
/*  ----- quadratic del ----- */
dels  = (( p * p * p ) + ( 1.5 * p * p ) + ( 0.5 * p )) / ( 3 * maxkT );
for (i = 0; i <= p + 1; i++)
       del[i] = kb * T * (i+1) * (i+1) / dels;
break;

case 3:
/*  ----- cubic del -----  */
dels = ( p * p ) * ( p * p + 2 * p + 1) / ( 4 * maxkT);
for (i = 0; i <= p + 1; i++)
       del[i] = kb * T * (i+1) * (i+1) * (i+1) / dels;
break;

case 4:
/*  ----- Linear del ----- */
    for (i = 0; i <= p; i++)
    {
        if ( i <= 50)
            del[i] = kb * T / 100;
        else
            del[i] =  kb * T / 100;

    }
break;

case 5:
/*  ----- polariton del -----   */
/*  dk quadratic in k */
dels  = (( p * p * p ) + ( 1.5 * p * p ) + ( 0.5 * p )) /  (3*maxkp);
if(disp == 1)
{
kpp[0] = 0;
f[0][0] = 0;
for(i = 1; i <= p + 2; i++)
{
kpp[i] = kpp[i-1] + delk[i-1];
f[i][0]  = hb * hb * eC * kpp[i] * kpp[i] / (2 * em * (Me + Mh));
}
}
if(disp == 2 || disp == 3)
{
    kpp[0] = 0;
    Exk[0] = Ex0;
    Eck[0] = hb * vc * kcz / indexr;
    Elp1[0] = Exk[0] * Exk[0] + Eck[0] * Eck[0] + gs * gs;
    f[0][0] = sqrt((Elp1[0] - sqrt(Elp1[0] * Elp1[0] - 4 * Exk[0] * Exk[0] * Eck[0] * Eck[0])) / 2);
    f[0][4] = f[0][0];
    delk[0] = 1 / dels;
    for (i = 1; i <= p + 2; i++)
    {
        delk[i] = (1+i)*(1+i) / dels;
        kpp[i] = kpp[i-1] + delk[i-1];
        Exk[i] = Ex0 + (hb * hb * kpp[i] * kpp[i] * eC) / (2 * em * (Me + Mh));
        Eck[i] = hb * (vc / indexr) * sqrt((kcz * kcz) + (kpp[i] * kpp[i]));
        Elp1[i] = Exk[i] * Exk[i] + Eck[i] * Eck[i] + gs * gs;
        f[i][0] = sqrt((Elp1[i] - sqrt(Elp1[i] * Elp1[i] - 4 * Exk[i] * Exk[i] * Eck[i] * Eck[i])) / 2);
        f[i][4] = f[i][0];
        f[i][5] = f[i][0];
    }
```

```
}
for (i = 0; i <= p + 1; i++)
{
     del[i] = f[i+1][0] - f[i][0];
     dEdki[i] = getdEdk(f[i][0], kpp[i], disp, indexr, i, 1);
}
/* for free electrons */
DOSe[0] = 2 * em * Me / (pi * hb * hb * eC);
Ne[0] = 0;
qo[0] = 0; /*eC / (2 * einf * eo);*/  /* added (2 * pi) on 3/27/08 */
for(i = 0; i <= p+1; i++)
{
     Eek[i] = hb * hb * kpp[i] * kpp[i] * eC / (2 * em * Me);
     fe[i] = 1 / (exp( (Eek[i] / (kb * TTT[0])) - (mue[0]) ) + 1);
}
for(i = 0; i <= p; i++)
     Ne[0] += fe[i] * DOSe[0] * (Eek[i+1] - Eek[i]);
/*printf("Ne = %e\n", Ne[0]);
exit(1);  */
break;

}
}
```

B.10 GETF4.H

```
double Determinef4(double E4, double KK, double f[1000][6], int m, int lll,



int zzz, int Geo, double fpp[1000], double delk[1000], double del[1000])
{
        if ( lll == -1)
        {
           if (E4 < f[p+1][0])
                {
                while ( E4 > f[zzz][0])
                {
                        zzz++;
                        if (zzz > p + 1)
                            zzz = 0;
                }
                zzz--;
                if( (E4 < f[zzz][0]) || (E4 > f[zzz+1][0]) )
                {
                     printf("f4 error\n");
                     printf("E4 = %.12f, E(z) = %f, z = %d, point = %f\n", E4, f[zzz][0], zzz, point);
                     exit(1);
                }
                point = (E4-f[zzz][0])/ del[zzz];
                ff4 = f[zzz][1] + (point * (f[zzz+1][1] - f[zzz][1]));
                }
           else
                ff4 = 0;
        }

        else
        {
            if ( lll == m)
                point = 0;
            else
            {
                if ( Geo == 5)
                    point = (KK - kpp[lll]) / delk[lll];
                else
                    point = (E4-f[lll][0])/ del[lll];
            }
            ff4 = f[lll][1] + (point * (f[lll+1][1] - f[lll][1]));

        }
return(ff4);
}
```

B.11 POLCHECK.H

```
void polprint(double kpp[1000], long double f[1000][4], double DOS[1000], double N[1000], double indexr, int y)
{
    double dddd;
    printf("i  k  E  DOS  dE/dk  N\n");
    for(i=0;i<=p;i++)
    {
         printf("%d %.6e  ", i, kpp[i]);
         printf("%e  ", f[i][0]);
         printf("%f  ", DOS[i]);
         dddd = getdEdk(f[i][0], kpp[i], disp, indexr, i, y);
         printf("%e  ", dddd);
         printf("%e\n", N[i]);
    }
}
```

B.12 POLENERGY.H

```
double polE(double kpp1, double index, int disp)
{
if(disp == 1)
    Elpt = hb * hb * kpp1 * kpp1 * eC / (2 * em * (Me + Mh));

if(disp == 2)
{
    Exkt = Ex0 + (hb * hb * kpp1 * kpp1 * eC) / (2 * em * (Me + Mh));
    Eckt = hb * vc * sqrt((kcz * kcz) + (kpp1 * kpp1)) / index;
    Elp1t = Exkt * Exkt + Eckt * Eckt + gs * gs;
    Elpt = sqrt((Elp1t - sqrt((Elp1t * Elp1t) - (4 * Exkt * Exkt * Eckt * Eckt))) / 2);
}
return(Elpt);
}
```

B.13 POLFRACTION.H

```
int polfrac()
{
for(i = 0; i <= p; i++)
{
    if(disp == 1)
 Xp[i] = 1;
    if(disp == 2 || disp == 3)
 Xp[i] = (((Eck[i] - Exk[i]) / sqrt(Elp1[i] - 2 * Eck[i] * Exk[i])) + 1) / 2;
    ki[i] = (kpp[i] + kpp[i+1]) / 2;
 /* printf("Xp%d = %e\n", i, Xp[i]); */
}
return(1);
}
```

B.14 PULSEFLAT.H

```
int pulseflatf(double del[1000], int p, double DOS[1000], double f[1000][6], double N[1000], double expp, double indexr, int y)
{
```

```
delSet(del, expp, delc, f, indexr);

for(i = 0; i <= p; i++)
{
    for(j = 0; j <= p; j++)
        deldel[i][j] = del[i] * del[j];
}
if(Geo != 5)
{
    if (Geo == 4)
        f[0][0] = hw;
    else
        f[0][0] = 0;
    for (j = 1; j <= p + 1; j++)
        f[j][0] = f[j-1][0] + del[j-1];
}
for (j = 0; j <= p; j++)
     f[j][1] = (4E-15/pulseT)*g;
f[p+1][1] = f[p][1] * exp(-del[p] / (kb * T));
f[p+2][1] = f[p+1][1] * exp(-del[p+1] / (kb * T));

DOSf(DOS, f, indexr, y);

for (j = 0; j <= p+1; j++)
{
     f[j][2] = (DOS[j] + DOS[j+1]) * del[j] / 2;
     N[j] = f[j][1] * f[j][2];
}
N[p+3] = (4E-15 / pulseT) * DOS[p+3] * g;
f[p+3][1] = N[p+3] / DOS[p+3];

return(1);
}
```

B.15 UPDATEF.H

```
double updatef(double Ntotal, double uprate[1], double x,
double dN[1000], double N[1000], double f[1000][6], double Nin[1000], double
Nout[1000], int Geo, double kpp[1000], double Stime[200], double taut, double Avetau[0])
{
        double Nnewtau, Nnewtau1; /* for calculating average total lifetime */
        double deltaN;
        deltaN = 0;
        Ntotalin = 0;
        Ntotalout = 0;
        for (j = 0; j <= p; j++)
        {
             Ntotalin += Nin[j];
             Ntotalout += Nout[j];
        }
        /* printf("Ntotalin %e  Ntotalout %e Ntotal %e  uprate %e\n", Ntotalin, Ntotalout, Ntotal, uprate[0]); */

        x = uprate[0] * Ntotal;
        for (j = 0; j <= p; j++)
        {
             f[j][3] = (Nin[j]/Ntotalin) - (Nout[j]/Ntotalout);
             /* printf ("N%d = %e changes by %e with Nin%d = %e and Nout%d = %e\n", j, N[j], x * f[j][3], j, Nin[j], j, Nout[j]); */
     N[j] += x * f[j][3];
             deltaN += f[j][3];
             if(N[j] < 0)
     {
 N[j] -= x * f[j][3];
 N[j] /= 2;
     }
        }
        deltaNAll[0] += deltaN;
if(N[p+3] < 0) N[p+3] = 0;
        Stime[p+3] = 0;
        for (j = 0; j <= p; j++)
{
       if(Nin[j] != Nout[j]*Ntotalin/Ntotalout)
     {
```

```
                Stime[j] = f[j][3] / (Nin[j] - (Nout[j]*Ntotalin/Ntotalout));
         Stime[p+3] += fabs(Stime[j]);
     }
}
Stime[p+3] *= x / (p+1);
/* Stime[p+3] = 0;
        for (j = 0; j <= p; j++)
{
    if(Nin[j] != Nout[j])
    {
                Stime[p+3] += fabs(f[j][3]);
                Stime[j] = f[j][3] / (Nin[j] - Nout[j]);
       }
}
Stime[p+3] *= x  / fabs(Ntotalin - Ntotalout);*/

/* system losses */
        Nnewtau = 0;
        Nnewtau1 = 0;
        for(j = 0; j <= p; j++)
        {
            if(f[j][0] - f[0][0] < 0.0035)  Nnewtau1 += N[j];
            N[j] -= Stime[p+3] * N[j] * ( (Itaux * Xp[j]) + (Itaup * (1 - Xp[j])) );
/*N[j] -=  Stime[p+3] * Itau * N[j] / Xp[j];*/ /* *Xp[j]*Xp[j]*Xp[j]*Xp[j]*Xp[j]*Xp[j]);*/ /*Xp? see 7/10/06 in notes.




Should depend on photon fraction.  Why Xp^7?*/
            f[j][1] = N[j] / f[j][2];
            if(f[j][0] - f[0][0] < 0.0035) Nnewtau += N[j];
}
        Avetau[0] = (Stime[p+3] * Nnewtau1) / (Nnewtau1 - Nnewtau);
        /*printf("particle loss = %e \n", Stime[p+3] * Itau);*/
        N[p+3] -= Stime[p+3] * ( (Itaux * Xp[j]) + (Itaup * ( 1 - Xp[j])) ) * N[p+3]; /* 2^7 = 128 */
if(N[p+3] < 0) N[p+3] = 0;
f[p+3][1] = f[0][1]; /*N[p+3] / DOS[p+3];*/

if(initial == 7 && (taut + Stime[p+3]) < pulseT) /* puts in a boltzmann distribution at a given temperature */
{
for(j = 0; j <= p; j++)
{
       f[j][1] += (50 * Stime[p+3] / pulseT) / (exp(  ( ((kpp[j] * kpp[j] * hb * hb * eC) / (2 * em * (Me + Mh))) - mu) / (kb * TT) ) - 1);



/* set to put in thermalized excitons */
                                            /* (exp(  (f[j][0] - mu - f[0][0]) / (kb * TT) ) - 1); */
N[j] = f[j][1] * f[j][2];
}
N[p+3] += ( 50 * Stime[p+3] / pulseT) / (exp( - mu / (kb * TT) ) - 1);
f[p+3][1] = N[p+3] / DOS[p+3];
}
if(initial == 8 && (taut + Stime[p+3]) < pulseT) /* puts in a gaussian distribution at a specific energy */
{
for(j = 0; j <= p; j++)
{
f[j][1] += (Stime[p+3] / pulseT) * No[j] * 2 / (DOS[j] + DOS[j+1]);
N[j] = f[j][1] * f[j][2];
}
f[p+3][1] += (Stime[p+3] / pulseT) * No[0] * 2 / (DOS[0] + DOS[1]);
N[p+3] = f[p+3][1] * DOS[p+3];
}
                if(initial == 9 && (taut + Stime[p+3]) < pulseT) /* puts in a flat distribution in k space */
                {
                        for(j = 0; j <= p; j++)
                        {
                                f[j][1] += (Stime[p+3]/ pulseT) * g;
                                N[j] = f[j][1] * f[j][2];
                        }
                        f[p+3][1] += (Stime[p+3] /pulseT) * g;
                        N[p+3] = f[p+3][1] * DOS[p+3];
                }
         f[p+1][1] = f[p][1] * exp( -del[p] / (kb * T) );
 f[p+2][1] = f[p+1][1] * exp( -del[p+1] / (kb * T) );
         /*printf("f(0) = %e\n", f[0][1]);
         exit(1); */
return(x);
}
```

B.16 POLPMATX.H

```
int polelMatx(double f[1000][6], double Ninpolel[1000], double Noutpolel[1000], int m,
double del[1000], int zzz, int Geo, double kpp[1000], double delk[1000], double index, int disp, int y)
{
double Fin0, Fout0, Fin00, Fout00, dEdk3, den00, rado, dEdki2, dEdki3, kmk1, kmk12, kxk, k1xk1, qo2;
nnn=0;
/* include a loop for each integration variable in this list */
/* E, E1, E2 */
/* Allconst1 = 6 * 196 * EB * EB * aB * aB / (hb * 32 * pi * pi * pi); */ /* eC * eC / (128 * hb * eo * eo * einf * einf * pi * pi * pi); */
Allconst1 = 1000 * EB * EB * aB * aB / (hb * pi * pi * pi * pi * pi * pi * pi);
qo2 = qo[0] * qo[0] * Ne[0] * Ne[0] / (kb * kb * TTT[0] * TTT[0]);
/*
Be = Me / (Me + Mh1);
Bh = Mh1 / (Me + Mh1);
*/

den00 = 0;
if (y == 1 || disp == 3)
{
     for(i = 0; i <= p; i++)
     {
          if(i == 0)
              kxk = 1;
          else
              kxk = kpp[i] * kpp[i];
          for(ii = 0; ii <= p; ii++)
          {
               if(i == 0 && ii != 0)
                   kxk1 = kpp[ii];
               else if(i != 0 && ii == 0)
                   kxk1 = kpp[i];
               else if(i == 0 && ii == 0)
                   kxk1 = 1;
               else kxk1 = kpp[i] * kpp[ii];
               if(ii == 0)
                   k1xk1 = 1;
               else
                   k1xk1 = kpp[ii] * kpp[ii];
               dEdki2 = dEdki[ii];
               for(iii = 0; iii <= p; iii++)
               {
                    E3 = f[ii][0] - f[i][0] + Eek[iii];
                    dEdki3 = dg / (4 * pi * DOSe[0]);   /* DOS is a constant for the electron */
                    if (E3 > Eek[0] && E3 < Eek[p+1])
                    {
                         k3 = sqrt( (2 * em * Me * (E3 - Eek[0]) ) / (hb * hb * eC) );
                         jj = 0;
                         while (kpp[jj] <= k3)
                              jj++;
                         I[i][ii][iii] = jj--;
                         k2k3 = kpp[iii] * k3;
                         /*   if (kpp[I[i][ii][iii]] > k3)
                         {
                              printf("error in polelMatx.h with I[i][ii][iii]\n");
                              printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                              printf("kpp - k3 = %e\n", kpp[I[i][ii][iii]] - k3);
                              printf("k3 = %e\n", k3);
                              exit(1);
                         }  */
 dEdk3 = dg / (4 * pi * DOSe[0]);
                         fnew[i][ii][iii] = (k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]];
/* take out k = 0 'point'                          if (i == 0)
 {
     E40[ii][iii] = f[iii][0] + f[ii][0] - f[0][0];
     k40[ii][iii] = Determinek3(E40[ii][iii], f, zzz);
     if( (k40[ii][iii] < (kpp[ii] + kpp[iii])) && ((k40[ii][iii]*k40[ii][iii]) > (kpp[ii]-kpp[iii])*(kpp[ii]-kpp[iii])) )
     {
          dEdk40[ii][iii] = getdEdk(E40[ii][iii], k40[ii][iii], disp, index, zzz, y);
  dfac0[ii][iii] = (0.5 * Xp[ii] * Xp[iii] * Xp[I[i][ii][iii]]) * deldel[ii][iii] * 2 * pi /



(dEdki2 * dEdki3 * dEdk40[ii][iii] * (sqrt((2 * kpp[iii] * kpp[iii] * kpp[ii] * kpp[ii]) - (k40[ii][iii]*k40[ii][iii]*k40[ii][iii]*k40[ii][iii]) +



(2*k40[ii][iii]*k40[ii][iii]*(kpp[ii]*kpp[ii] + kpp[iii]*kpp[iii])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) -
```

```
(kpp[iii]*kpp[iii]*kpp[iii]*kpp[iii])) ) );
                          }
     else dfac0[ii][iii] = 0;
 }
 if (iii == 0)
 {
      E400[i][ii] = f[i][0] + f[ii][0] - f[0][0];
      k400[i][ii] = Determinek3(E400[i][ii], f, zzz);
      if( (k400[i][ii] < (kpp[i] + kpp[ii])) && ((k400[i][ii]*k400[i][ii]) > (kpp[i]-kpp[ii])*(kpp[i]-kpp[ii])) )
      {
           dEdk400[i][ii] = getdEdk(E400[i][ii], k400[i][ii], disp, index, zzz, y);
           den00 = (2 * kpp[i] * kpp[i] * kpp[ii] * kpp[ii]) - (k400[i][ii] * k400[i][ii] * k400[i][ii] * k400[i][ii])



+ (2 * k400[i][ii] * k400[i][ii] * (kpp[ii]*kpp[ii] + kpp[i]*kpp[i])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) - (kpp[i]*kpp[i]*kpp[i]*kpp[i]);
           dfac00[i][ii] = 2 * (Xp[ii] * 0.5 * Xp[I[i][ii][iii]]) * del[ii] * 8 * pi * pi / (D2A * dEdki2 * dEdk400[i][ii] * sqrt(den00) );
                           }
      else dfac00[i][ii] = 0;
 }    */
 radp = (kxk - kpp[iii] * kpp[iii] - k2k3) / kxk1;
                         radn = (kxk - kpp[iii] * kpp[iii] + k2k3) / kxk1;
                         if(radp < -1)
                              thetamax = pi;
                         else if(radp > 1)
                              thetamax = 0;
                         else
                              thetamax = acos(radp);
                         if (radn > 1)
                              thetamin = 0;
                         else if (radn < -1)
                              thetamin = pi;
                         else
                              thetamin = acos(radn);
                         if (thetamax - thetamin > 1e-3) /* 1e-13 */
                         {
                             dif = thetamax - thetamin;
                             ttheta = thetamax + thetamin;
                             GQ1 = 0;
                             for(j = 1; j <= GQp; j++)
                             {
                                  rado = cos(((dif * yy[j]) + ttheta) / 2);
                                  kmk12 = kxk + k1xk1 - (2 * kxk1 * rado);
                                  /* if(kmk12 < 0) kmk12 = 0; */

/*
                                  kmk1 = sqrt(kmk12);
                                  Bhka = Bh * kmk1 * aB / 2; */ /* variiertes sigma_h */
                                  Beka = kxk * aB * aB / 4;
                                  Bcka = kmk12 * aB * aB;
                                  Bhka3 = (2 + Beka) * (2 + Beka) * (2 + Beka);
                                  Bhka34 = (2 + Bcka + Beka) * (2 + Bcka + Beka) * (2 + Bcka + Beka);
                                  Beka3 = (k3 * k3) + (1 / (aB * aB)) + kmk12;
                                  Behka = Beka3 + qo2 + (2 * sqrt(qo2 * Beka3));
/*
                                  Bcka3 = 1 / ( (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) );
                                  Bhka3Beka3 = Bhka3 + Beka3 - Behka;
                                  if (fabs(Bhka3Beka3) < 1e-15)  Bhka3Beka3 = 0;
                                  Bcka3Bhka34 = Bcka3 + Bhka34 - (2 * sqrt(Bcka3 * Bhka34));
                                  if (fabs(Bcka3Bhka34) < 1e-15) Bcka3Bhka34 = 0;
*/
                                  if(ScreenType = 1)
                                       GQ1 += ww[j] / ( ((1 / (kmk12 * aB * aB)) * (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) *



sqrt( (radp - rado) * (-radn + rado) ) );
                                  else
                                       GQ1 += ww[j] / ( Bhka3 * Bhka34 * Behka * sqrt( (radp - rado) * (-radn + rado) ) );
                                  /* GQ1 += (ww[j] / sqrt( (radp - rado) * (-radn + rado) )) *



( (Bhka3Beka3 / (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) +
```

```
(80 * exp(-2 * aB * aB * (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12))))) ); */
/*
GQ1 += ww[j] * (  (Bhka3Beka3 + (80 * exp(-2 * kmk12 * aB * aB)) )  /



( qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))  ) / sqrt( (radp - rado) * (-radn + rado) );*/



/* 3/21/08   removed (16 * Bcka3Bhka34 / ((1/(aB * aB)) + qo2))  ) with ( 225 * exp(-(kmk12 * aB * aB)) / qo2)  */



/* changed 225 to 80 = (14 * 2 / pi)^2 */
                            }
                            GQ1 *= dif /(2 * kxk1);
                            dfacel1[i][ii][iii] = Allconst1 * Xp[ii] * GQ1 * del[ii] * (Eek[iii+1] - Eek[iii]) / (dEdki2 * dEdki3 * dEdk3);
                        }
                        else
                            dfacel1[i][ii][iii] = 0;
                    }
                    else
                    {
                          dfacel1[i][ii][iii] = 0;
                          if(i == 0) dfacel0[ii][iii] = 0;
                    }
                    if (dfacel0[ii][iii] != dfacel0[ii][iii] || dfacel0[ii][iii] > 1e60 || dfacel00[i][ii] != dfacel00[i][ii] ||



 dfacel00[i][ii] > 1e60 || dfacel0[ii][iii] < -1e60 || dfacel00[i][ii] < -1e60 || dfacel1[i][ii][iii] < - 1e60 ||



 dfacel1[i][ii][iii] > 1e60 || dfacel1[i][ii][iii] != dfacel1[i][ii][iii])
                    {
                        printf("error (2) in polelMatx.h\n");
                        printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                        printf("df1 =  %e     df0 = %e     df00 = %e    \n", dfacel1[i][ii][iii], dfacel0[ii][iii], dfacel00[i][ii]);
               printf("GQ = %e, dEdk2 =%e, dEdk3 = %e, dEdk4 = %e, dif = %e\n", GQ1, dEdki2, dEdki3, dEdk3, dif);
                        printf("E400 =  %e     k400 = %e     den00 = %e    \n", E400[i][ii], k400[i][ii], den00);
                        printf("dEdk400 = %e, kxk1 = %e\n", dEdk400[i][ii], kxk1);
                        exit(1);
                    }
    if(den00 < 0)
    {
         printf("den00 error in polelMatx.h, k = %e, k1 = %e,k400 = %e, den00 = %e\n", kpp[i], kpp[ii], k400[i][ii], den00);
 printf("i = %d, ii = %d\n", i, ii);
                         exit(1);
    }
/*               printf("dfac = %e\n", dfacel1[i][ii][iii]);   */
              }
          }
     }
}
Ninpolel[p+3] = 0;
Noutpolel[p+3] = 0;
return(1);
}
```

B.17 POLELMATX.H

```
int polelMatx(double f[1000][6], double Ninpolel[1000], double Noutpolel[1000], int m,
double del[1000], int zzz, int Geo, double kpp[1000], double delk[1000], double index, int disp, int y)
{
double Fin0, Fout0, Fin00, Fout00, dEdk3, den00, rado, dEdki2, dEdki3, kmk1, kmk12, kxk, k1xk1, qo2;
nnn=0;
/* include a loop for each integration variable in this list */
/* E, E1, E2 */
/* Allconst1 = 6 * 196 * EB * EB * aB * aB / (hb * 32 * pi * pi * pi); */ /* eC * eC / (128 * hb * eo * eo * einf * einf * pi * pi * pi); */
```

```
Allconst1 = 1000 * EB * EB * aB * aB / (hb * pi * pi * pi * pi * pi * pi);
qo2 = qo[0] * qo[0] * Ne[0] * Ne[0] / (kb * kb * TTT[0] * TTT[0]);
/*
Be = Me / (Me + Mh1);
Bh = Mh1 / (Me + Mh1);
*/

den00 = 0;
if (y == 1 || disp == 3)
{
     for(i = 0; i <= p; i++)
     {
          if(i == 0)
              kxk = 1;
          else
              kxk = kpp[i] * kpp[i];
          for(ii = 0; ii <= p; ii++)
          {
               if(i == 0 && ii != 0)
                   kxk1 = kpp[ii];
               else if(i != 0 && ii == 0)
                   kxk1 = kpp[i];
               else if(i == 0 && ii == 0)
                   kxk1 = 1;
               else kxk1 = kpp[i] * kpp[ii];
               if(ii == 0)
                   k1xk1 = 1;
               else
                   k1xk1 = kpp[ii] * kpp[ii];
               dEdki2 = dEdki[ii];
               for(iii = 0; iii <= p; iii++)
               {
                    E3 = f[ii][0] - f[i][0] + Eek[iii];
                    dEdki3 = dg / (4 * pi * DOSe[0]);   /* DOS is a constant for the electron */
                    if (E3 > Eek[0] && E3 < Eek[p+1])
                    {
                         k3 = sqrt( (2 * em * Me * (E3 - Eek[0]) ) / (hb * hb * eC) );
                         jj = 0;
                         while (kpp[jj] <= k3)
                              jj++;
                         I[i][ii][iii] = jj--;
                         k2k3 = kpp[iii] * k3;
                         /*   if (kpp[I[i][ii][iii]] > k3)
                         {
                              printf("error in polelMatx.h with I[i][ii][iii]\n");
                              printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                              printf("kpp - k3 = %e\n", kpp[I[i][ii][iii]] - k3);
                              printf("k3 = %e\n", k3);
                              exit(1);
                         }  */
 dEdk3 = dg / (4 * pi * DOSe[0]);
                         fnew[i][ii][iii] = (k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]];
/* take out k = 0 'point'                          if (i == 0)
 {
     E40[ii][iii] = f[iii][0] + f[ii][0] - f[0][0];
     k40[ii][iii] = Determinek3(E40[ii][iii], f, zzz);
     if( (k40[ii][iii] < (kpp[ii] + kpp[iii])) && ((k40[ii][iii]*k40[ii][iii]) > (kpp[ii]-kpp[iii])*(kpp[ii]-kpp[iii])) )
     {
          dEdk40[ii][iii] = getdEdk(E40[ii][iii], k40[ii][iii], disp, index, zzz, y);
  dfac0[ii][iii] = (0.5 * Xp[ii] * Xp[iii] * Xp[I[i][ii][iii]]) * deldel[ii][iii] * 2 * pi /



(dEdki2 * dEdki3 * dEdk40[ii][iii] * (sqrt((2 * kpp[iii] * kpp[iii] * kpp[ii] * kpp[ii]) - (k40[ii][iii]*k40[ii][iii]*k40[ii][iii]*k40[ii][iii]) +



(2*k40[ii][iii]*k40[ii][iii]*(kpp[ii]*kpp[ii] + kpp[iii]*kpp[iii])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) -



(kpp[iii]*kpp[iii]*kpp[iii]*kpp[iii])) ) );
                              }
     else dfac0[ii][iii] = 0;
 }
 if (iii == 0)
 {
      E400[i][ii] = f[i][0] + f[ii][0] - f[0][0];
      k400[i][ii] = Determinek3(E400[i][ii], f, zzz);
```

```
      if( (k400[i][ii] < (kpp[i] + kpp[ii])) && ((k400[i][ii]*k400[i][ii]) > (kpp[i]-kpp[ii])*(kpp[i]-kpp[ii])) )
      {
           dEdk400[i][ii] = getdEdk(E400[i][ii], k400[i][ii], disp, index, zzz, y);
           den00 = (2 * kpp[i] * kpp[i] * kpp[ii] * kpp[ii]) -



(k400[i][ii] * k400[i][ii] * k400[i][ii] * k400[i][ii]) + (2 * k400[i][ii] * k400[i][ii] *



(kpp[ii]*kpp[ii] + kpp[i]*kpp[i])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) - (kpp[i]*kpp[i]*kpp[i]*kpp[i]);
           dfac00[i][ii] = 2 * (Xp[ii] * 0.5 * Xp[I[i][ii][iii]]) * del[ii] * 8 * pi * pi / (D2A * dEdki2 * dEdk400[i][ii] * sqrt(den00) );
                           }
      else dfac00[i][ii] = 0;
 }    */
 radp = (kxk - kpp[iii] * kpp[iii] - k2k3) / kxk1;
                         radn = (kxk - kpp[iii] * kpp[iii] + k2k3) / kxk1;
                         if(radp < -1)
                              thetamax = pi;
                         else if(radp > 1)
                              thetamax = 0;
                         else
                              thetamax = acos(radp);
                         if (radn > 1)
                              thetamin = 0;
                         else if (radn < -1)
                              thetamin = pi;
                         else
                              thetamin = acos(radn);
                         if (thetamax - thetamin > 1e-3) /* 1e-13 */
                         {
                             dif = thetamax - thetamin;
                             ttheta = thetamax + thetamin;
                             GQ1 = 0;
                             for(j = 1; j <= GQp; j++)
                             {
                                  rado = cos(((dif * yy[j]) + ttheta) / 2);
                                  kmk12 = kxk + k1xk1 - (2 * kxk1 * rado);
                                  /* if(kmk12 < 0) kmk12 = 0; */

/*
                                  kmk1 = sqrt(kmk12);
                                  Bhka = Bh * kmk1 * aB / 2; */ /* variiertes sigma_h */
                                  Beka = kxk * aB * aB / 4;
                                  Bcka = kmk12 * aB * aB;
                                  Bhka3 = (2 + Beka) * (2 + Beka) * (2 + Beka);
                                  Bhka34 = (2 + Bcka + Beka) * (2 + Bcka + Beka) * (2 + Bcka + Beka);
                                  Beka3 = (k3 * k3) + (1 / (aB * aB)) + kmk12;
                                  Behka = Beka3 + qo2 + (2 * sqrt(qo2 * Beka3));
/*
                                  Bcka3 = 1 / ( (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) );
                                  Bhka3Beka3 = Bhka3 + Beka3 - Behka;
                                  if (fabs(Bhka3Beka3) < 1e-15)  Bhka3Beka3 = 0;
                                  Bcka3Bhka34 = Bcka3 + Bhka34 - (2 * sqrt(Bcka3 * Bhka34));
                                  if (fabs(Bcka3Bhka34) < 1e-15) Bcka3Bhka34 = 0;
*/
                                  if(ScreenType = 1)
                                       GQ1 += ww[j] / ( ((1 / (kmk12 * aB * aB)) *



(qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) * sqrt( (radp - rado) * (-radn + rado) ) );
                                  else
                                       GQ1 += ww[j] / ( Bhka3 * Bhka34 * Behka * sqrt( (radp - rado) * (-radn + rado) ) );
                                  /* GQ1 += (ww[j] / sqrt( (radp - rado) * (-radn + rado) )) *



( (Bhka3Beka3 / (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) + (80 * exp(-2 * aB * aB * (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12))))) ); */
/*
GQ1 += ww[j] * (  (Bhka3Beka3 + (80 * exp(-2 * kmk12 * aB * aB)) )  /



( qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))  ) / sqrt( (radp - rado) * (-radn + rado) );*/
```

```
/* 3/21/08   removed (16 * Bcka3Bhka34 / ((1/(aB * aB)) + qo2))  ) with ( 225 * exp(-(kmk12 * aB * aB)) / qo2)  */



 /* changed 225 to 80 = (14 * 2 / pi)^2 */
                              }
                              GQ1 *= dif /(2 * kxk1);
                              dfacel1[i][ii][iii] = Allconst1 * Xp[ii] * GQ1 * del[ii] * (Eek[iii+1] - Eek[iii]) / (dEdki2 * dEdki3 * dEdk3);
                         }
                         else
                              dfacel1[i][ii][iii] = 0;
                    }
                    else
                    {
                          dfacel1[i][ii][iii] = 0;
                          if(i == 0) dfacel0[ii][iii] = 0;
                    }
                    if (dfacel0[ii][iii] != dfacel0[ii][iii] || dfacel0[ii][iii] > 1e60 || dfacel00[i][ii] !=



dfacel00[i][ii] || dfacel00[i][ii] > 1e60 || dfacel0[ii][iii] < -1e60 || dfacel00[i][ii] < -1e60 ||



dfacel1[i][ii][iii] < - 1e60 || dfacel1[i][ii][iii] > 1e60 || dfacel1[i][ii][iii] != dfacel1[i][ii][iii])
                    {
                        printf("error (2) in polelMatx.h\n");
                        printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                        printf("df1 =  %e     df0 = %e     df00 = %e    \n", dfacel1[i][ii][iii], dfacel0[ii][iii], dfacel00[i][ii]);
               printf("GQ = %e, dEdk2 =%e, dEdk3 = %e, dEdk4 = %e, dif = %e\n", GQ1, dEdki2, dEdki3, dEdk3, dif);
                        printf("E400 =  %e     k400 = %e     den00 = %e    \n", E400[i][ii], k400[i][ii], den00);
                        printf("dEdk400 = %e, kxk1 = %e\n", dEdk400[i][ii], kxk1);
                        exit(1);
                    }
    if(den00 < 0)
    {
         printf("den00 error in polelMatx.h, k = %e, k1 = %e,k400 = %e, den00 = %e\n", kpp[i], kpp[ii], k400[i][ii], den00);
 printf("i = %d, ii = %d\n", i, ii);
                         exit(1);
    }
/*               printf("dfac = %e\n", dfacel1[i][ii][iii]);   */
              }
          }
     }
}
Ninpolel[p+3] = 0;
Noutpolel[p+3] = 0;
return(1);
}
```

B.18 POLPTAMATX.H

```
int polelMatx(double f[1000][6], double Ninpolel[1000], double Noutpolel[1000], int m,
double del[1000], int zzz, int Geo, double kpp[1000], double delk[1000], double index, int disp, int y)
{
double Fin0, Fout0, Fin00, Fout00, dEdk3, den00, rado, dEdki2, dEdki3, kmk1, kmk12, kxk, k1xk1, qo2;
nnn=0;
/* include a loop for each integration variable in this list */
/* E, E1, E2 */
/* Allconst1 = 6 * 196 * EB * EB * aB * aB / (hb * 32 * pi * pi * pi); */ /* eC * eC / (128 * hb * eo * eo * einf * einf * pi * pi * pi); */
Allconst1 = 1000 * EB * EB * aB * aB / (hb * pi * pi * pi * pi * pi * pi);
qo2 = qo[0] * qo[0] * Ne[0] * Ne[0] / (kb * kb * TTT[0] * TTT[0]);
/*
Be = Me / (Me + Mh1);
Bh = Mh1 / (Me + Mh1);
*/

den00 = 0;
if (y == 1 || disp == 3)
{
     for(i = 0; i <= p; i++)
```

```
    {
         if(i == 0)
             kxk = 1;
         else
             kxk = kpp[i] * kpp[i];
         for(ii = 0; ii <= p; ii++)
         {
              if(i == 0 && ii != 0)
                  kxk1 = kpp[ii];
              else if(i != 0 && ii == 0)
                  kxk1 = kpp[i];
              else if(i == 0 && ii == 0)
                  kxk1 = 1;
              else kxk1 = kpp[i] * kpp[ii];
              if(ii == 0)
                  k1xk1 = 1;
              else
                  k1xk1 = kpp[ii] * kpp[ii];
              dEdki2 = dEdki[ii];
              for(iii = 0; iii <= p; iii++)
              {
                   E3 = f[ii][0] - f[i][0] + Eek[iii];
                   dEdki3 = dg / (4 * pi * DOSe[0]);   /* DOS is a constant for the electron */
                   if (E3 > Eek[0] && E3 < Eek[p+1])
                   {
                        k3 = sqrt( (2 * em * Me * (E3 - Eek[0]) ) / (hb * hb * eC) );
                        jj = 0;
                        while (kpp[jj] <= k3)
                             jj++;
                        I[i][ii][iii] = jj--;
                        k2k3 = kpp[iii] * k3;
                        /*   if (kpp[I[i][ii][iii]] > k3)
                        {
                             printf("error in polelMatx.h with I[i][ii][iii]\n");
                             printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                             printf("kpp - k3 = %e\n", kpp[I[i][ii][iii]] - k3);
                             printf("k3 = %e\n", k3);
                             exit(1);
                        }  */
 dEdk3 = dg / (4 * pi * DOSe[0]);
                        fnew[i][ii][iii] = (k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]];
/* take out k = 0 'point'                        if (i == 0)
 {
     E40[ii][iii] = f[iii][0] + f[ii][0] - f[0][0];
     k40[ii][iii] = Determinek3(E40[ii][iii], f, zzz);
     if( (k40[ii][iii] < (kpp[ii] + kpp[iii])) && ((k40[ii][iii]*k40[ii][iii]) > (kpp[ii]-kpp[iii])*(kpp[ii]-kpp[iii])) )
     {
          dEdk40[ii][iii] = getdEdk(E40[ii][iii], k40[ii][iii], disp, index, zzz, y);
  dfac0[ii][iii] = (0.5 * Xp[ii] * Xp[iii] * Xp[I[i][ii][iii]]) * deldel[ii][iii] * 2 * pi /



(dEdki2 * dEdki3 * dEdk40[ii][iii] * (sqrt((2 * kpp[iii] * kpp[iii] * kpp[ii] * kpp[ii]) - (k40[ii][iii]*k40[ii][iii]*k40[ii][iii]*k40[ii][iii]) +



(2*k40[ii][iii]*k40[ii][iii]*(kpp[ii]*kpp[ii] + kpp[iii]*kpp[iii])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) -



(kpp[iii]*kpp[iii]*kpp[iii]*kpp[iii])) ) );
                             }
     else dfac0[ii][iii] = 0;
 }
 if (iii == 0)
 {
      E400[i][ii] = f[i][0] + f[ii][0] - f[0][0];
      k400[i][ii] = Determinek3(E400[i][ii], f, zzz);
      if( (k400[i][ii] < (kpp[i] + kpp[ii])) && ((k400[i][ii]*k400[i][ii]) > (kpp[i]-kpp[ii])*(kpp[i]-kpp[ii])) )
      {
           dEdk400[i][ii] = getdEdk(E400[i][ii], k400[i][ii], disp, index, zzz, y);
           den00 = (2 * kpp[i] * kpp[i] * kpp[ii] * kpp[ii]) - (k400[i][ii] * k400[i][ii] * k400[i][ii] * k400[i][ii]) +



(2 * k400[i][ii] * k400[i][ii] * (kpp[ii]*kpp[ii] + kpp[i]*kpp[i])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) - (kpp[i]*kpp[i]*kpp[i]*kpp[i]);
           dfac00[i][ii] = 2 * (Xp[ii] * 0.5 * Xp[I[i][ii][iii]]) * del[ii] * 8 * pi * pi / (D2A * dEdki2 * dEdk400[i][ii] * sqrt(den00) );
                             }
      else dfac00[i][ii] = 0;
```

```
 }    */
 radp = (kxk - kpp[iii] * kpp[iii] - k2k3) / kxk1;
                         radn = (kxk - kpp[iii] * kpp[iii] + k2k3) / kxk1;
                         if(radp < -1)
                              thetamax = pi;
                         else if(radp > 1)
                              thetamax = 0;
                         else
                              thetamax = acos(radp);
                         if (radn > 1)
                              thetamin = 0;
                         else if (radn < -1)
                              thetamin = pi;
                         else
                              thetamin = acos(radn);
                         if (thetamax - thetamin > 1e-3) /* 1e-13 */
                         {
                             dif = thetamax - thetamin;
                             ttheta = thetamax + thetamin;
                             GQ1 = 0;
                             for(j = 1; j <= GQp; j++)
                             {
                                  rado = cos(((dif * yy[j]) + ttheta) / 2);
                                  kmk12 = kxk + k1xk1 - (2 * kxk1 * rado);
                                  /* if(kmk12 < 0) kmk12 = 0; */

/*
                                  kmk1 = sqrt(kmk12);
                                  Bhka = Bh * kmk1 * aB / 2; */ /* variiertes sigma_h */
                                  Beka = kxk * aB * aB / 4;
                                  Bcka = kmk12 * aB * aB;
                                  Bhka3 = (2 + Beka) * (2 + Beka) * (2 + Beka);
                                  Bhka34 = (2 + Bcka + Beka) * (2 + Bcka + Beka) * (2 + Bcka + Beka);
                                  Beka3 = (k3 * k3) + (1 / (aB * aB)) + kmk12;
                                  Behka = Beka3 + qo2 + (2 * sqrt(qo2 * Beka3));
/*
                                  Bcka3 = 1 / ( (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) * (1 + (Bcka * Bcka)) );
                                  Bhka3Beka3 = Bhka3 + Beka3 - Behka;
                                  if (fabs(Bhka3Beka3) < 1e-15)  Bhka3Beka3 = 0;
                                  Bcka3Bhka34 = Bcka3 + Bhka34 - (2 * sqrt(Bcka3 * Bhka34));
                                  if (fabs(Bcka3Bhka34) < 1e-15) Bcka3Bhka34 = 0;
*/
                                  if(ScreenType = 1)
                                       GQ1 += ww[j] / ( ((1 / (kmk12 * aB * aB)) *



(qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) * sqrt( (radp - rado) * (-radn + rado) ) );
                                  else
                                       GQ1 += ww[j] / ( Bhka3 * Bhka34 * Behka * sqrt( (radp - rado) * (-radn + rado) ) );
                                  /* GQ1 += (ww[j] / sqrt( (radp - rado) * (-radn + rado) )) *



( (Bhka3Beka3 / (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))) + (80 * exp(-2 * aB * aB * (qo2 + kmk12 + (2 * sqrt(qo2 * kmk12))))) ); */
/*
GQ1 += ww[j] * (  (Bhka3Beka3 + (80 * exp(-2 * kmk12 * aB * aB)) )  /



( qo2 + kmk12 + (2 * sqrt(qo2 * kmk12)))  ) / sqrt( (radp - rado) * (-radn + rado) );*/



/* 3/21/08   removed (16 * Bcka3Bhka34 / ((1/(aB * aB)) + qo2))  ) with ( 225 * exp(-(kmk12 * aB * aB)) / qo2)  */



/* changed 225 to 80 = (14 * 2 / pi)^2 */
                             }
                             GQ1 *= dif /(2 * kxk1);
                             dfacel1[i][ii][iii] = Allconst1 * Xp[ii] * GQ1 * del[ii] * (Eek[iii+1] - Eek[iii]) / (dEdki2 * dEdki3 * dEdk3);
                         }
                         else
                             dfacel1[i][ii][iii] = 0;
                    }
                    else
                    {
                          dfacel1[i][ii][iii] = 0;
```

```
                        if(i == 0) dfacel0[ii][iii] = 0;
                    }
                    if (dfacel0[ii][iii] != dfacel0[ii][iii] || dfacel0[ii][iii] > 1e60 || dfacel00[i][ii] != dfacel00[i][ii]


|| dfacel00[i][ii] > 1e60 || dfacel0[ii][iii] < -1e60 || dfacel00[i][ii] < -1e60 ||


 dfacel1[i][ii][iii] < - 1e60 || dfacel1[i][ii][iii] > 1e60 || dfacel1[i][ii][iii] != dfacel1[i][ii][iii])
                    {
                        printf("error (2) in polelMatx.h\n");
                        printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                        printf("df1 =  %e     df0 = %e     df00 = %e    \n", dfacel1[i][ii][iii], dfacel0[ii][iii], dfacel00[i][ii]);
               printf("GQ = %e, dEdk2 =%e, dEdk3 = %e, dEdk4 = %e, dif = %e\n", GQ1, dEdki2, dEdki3, dEdk3, dif);
                        printf("E400 =  %e     k400 = %e     den00 = %e    \n", E400[i][ii], k400[i][ii], den00);
                        printf("dEdk400 = %e, kxk1 = %e\n", dEdk400[i][ii], kxk1);
                        exit(1);
                    }
    if(den00 < 0)
    {
         printf("den00 error in polelMatx.h, k = %e, k1 = %e,k400 = %e, den00 = %e\n", kpp[i], kpp[ii], k400[i][ii], den00);
 printf("i = %d, ii = %d\n", i, ii);
                         exit(1);
    }
/*               printf("dfac = %e\n", dfacel1[i][ii][iii]);   */
              }
          }
     }
}
Ninpolel[p+3] = 0;
Noutpolel[p+3] = 0;
return(1);
}
```

B.19 POLPTA_PIEZOMATX.H

```
int MTAPiezo2(double W[200][200][200], double kpp[1000], double defe, double defh, double index, double kcz,



double W0TA[1000], double Wden0TA[1000], double Wden00TA[1000])
{
double Bp, E, E1, Ipe, Iph, Ipe1, Iph1, phi, phibad, q, q2, qz, Wden1TA;
double Ipe10, Iph10, Ipe0, Iph0;
long double E2;
double defepart, defhpart, piezopart1, piezopart2e, piezopart2h, Allpartse, Allpartsh, Allpartseh;
double phiq, qtot;

Wden1TA = 16 * pi * pi * vTA * vTA * vTA * hb * hb * Sd / eC;
for (j = 0; j <= p; j++)
{
    Xj = Xp[j];
    for (jj = 0; jj <= p; jj++)
    {
        Xjj = Xp[jj];
        NinkTA[j][jj] = 0;
E = f[j][0]; /*polE(kpp[j], index);*/
        E1 = f[jj][0]; /*polE(kpp[jj], index);*/
E2 = fabs(E - E1);  /* E - E1 */
        qtot = E2 / (hb * vTA);
phibad = ((kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (E2 * E2 / (hb * hb * vTA * vTA))) / (2 * kpp[j] * kpp[jj]);
if (fabs(phibad) < 1)
{
    phibad = acos(phibad);
    for (i = 1; i <= p; i++)
      {
phi = (i - 0.5) * phibad / p; /*(i - 0.5) * pi / p;*/
               phiq = atan( (kpp[jj] * sin(phi)) / (-kpp[j] + (kpp[jj] * cos(phi))) );
q2 = (kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (2 * kpp[j] * kpp[jj] * cos(phi));
```

```
/*(kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (2 * kpp[j] * kpp[jj] * cos(phi));*/
q = sqrt(q2);
Ipe1 = 1 + (Mh1 * q * aB / (2 * (Mh1 + Me))) * (Mh1 * q * aB / (2 * (Mh1 + Me)));
Iph1 = 1 + (Me * q * aB / (2 * (Mh1 + Me))) * (Me * q * aB / (2 * (Mh1 + Me)));
Ipe = 1 / (sqrt(Ipe1) * Ipe1);
Iph = 1 / (sqrt(Iph1) * Iph1);
if ( (((E2 * E2) / (hb * hb * vTA * vTA)) - q2) > 0)
{
       qz = sqrt(((E2 * E2) / (hb * hb * vTA * vTA)) - q2);
     Bp = 8 * pi * pi * sin(Lz * qz / 2) / (qz * Lz * ( (4 * pi * pi) - (Lz * Lz * qz * qz) ));
                    defepart = defe * defe * qtot;
                    defhpart = defh * defh * qtot;
                    piezopart1 = piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 / 8) ) /



(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                    /*
                    piezopart1 = 2 * piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 * cos(phiq) * cos(phiq) * sin(phiq) * sin(phiq)) ) /



(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                    */
        /*     piezopart2e = defe * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



( pi * einf * ( E2 * E2 / (hb * hb * vTA * vTA)) );
                    piezopart2h = defh * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



( pi * einf * ( E2 * E2 / (hb * hb * vTA * vTA)) ); */
                    piezopart2e = 0;
                    piezopart2h = 0;
                    Allpartse = (defepart + piezopart1 + piezopart2e) * Ipe * Ipe;
                    Allpartsh = (defhpart + piezopart1 + piezopart2h) * Iph * Iph;
                    Allpartseh = 2 * (-sqrt(defepart * defhpart) + piezopart1) * Ipe * Iph;

/****  W's assume area of cavity is 1 cm^2 *****/
     NinkTA[j][jj] +=  phibad * Bp * Bp * E2 * Xjj * Xj * (Allpartse + Allpartsh
+ Allpartseh ) / (p * qz);  /*Bp * Bp * hb * v * q * E2 * Xjj * Xj * (defe * Ipe - defh * Iph) * (defe * Ipe - defh * Iph) / qz; */
}
else
    NinkTA[j][jj] += 0;
phi = phibad + (( i - 0.5) * (pi - phibad) / p); /*(i - 0.5) * pi / p;*/
                phiq = atan( (kpp[jj] * sin(phi)) / (-kpp[j] + (kpp[jj] * cos(phi))) );
q2 = (kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (2 * kpp[j] * kpp[jj] * cos(phi));



/*(kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (2 * kpp[j] * kpp[jj] * cos(phi));*/
q = sqrt(q2);
Ipe1 = 1 + (Mh1 * q * aB / (2 * (Mh1 + Me))) * (Mh1 * q * aB / (2 * (Mh1 + Me)));
Iph1 = 1 + (Me * q * aB / (2 * (Mh1 + Me))) * (Me * q * aB / (2 * (Mh1  + Me)));
Ipe = 1 / (sqrt(Ipe1) * Ipe1);
Iph = 1 / (sqrt(Iph1) * Iph1);
if ( (((E2 * E2) / (hb * hb * vTA * vTA)) - q2) > 0)
{
     qz = sqrt(((E2 * E2) / (hb * hb * vTA * vTA)) - q2);
     Bp = 8 * pi * pi * sin(Lz * qz / 2) / (qz * Lz * ( (4 * pi * pi) - (Lz * Lz * qz * qz) ));
                    defepart = defe * defe * qtot;
                    defhpart = defh * defh * qtot;
                    piezopart1 = piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 / 8) )/



(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                    /*
                    piezopart1 = 2 * piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 * cos(phiq) * cos(phiq) * sin(phiq) *  sin(phiq)) ) /
```

```
(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                    */
             /* piezopart2e = defe * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



(pi * einf * ( E2 * E2 / (hb * hb * vTA * vTA)) );
                    piezopart2h = defh * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



(pi * einf * ( E2 * E2 / ( hb * hb * vTA * vTA)) ); */
                    piezopart2e = 0;
                    piezopart2h = 0;
                    Allpartse = (defepart + piezopart1 + piezopart2e) * Ipe * Ipe;
                    Allpartsh = (defhpart + piezopart1 + piezopart2h) * Iph * Iph;
                    Allpartseh = 2 * (-sqrt(defepart * defhpart) + piezopart1) * Ipe * Iph;
                /****  W's assume area of cavity is 1 cm^2 *****/
     NinkTA[j][jj] +=  (pi - phibad) * Bp * Bp * E2 * Xjj * Xj * (Allpartse + Allpartsh + Allpartseh) / (p * qz);
}
else
     NinkTA[j][jj] += 0;
     }
        }
else
{
     for (i = 1; i <= p; i++)
     {
                 phi = ((i - 0.5) * pi / p); /*(i - 0.5) * pi / p;*/
                 phiq = atan( (kpp[jj] * sin(phi)) / ( -kpp[j] + (kpp[jj] * cos(phi))) );
                 q2 = (kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) + (2 * kpp[j] * kpp[jj] * cos(phi));



/*(kpp[j] * kpp[j]) + (kpp[jj] * kpp[jj]) - (2 * kpp[j] * kpp[jj] * cos(phi));*/
                 q = sqrt(q2);
                 Ipe1 = 1 + (Mh1 * q * aB / (2 * (Mh1 + Me))) * (Mh1 * q * aB / (2 * (Mh1 + Me)));
                 Iph1 = 1 + (Me * q * aB / (2 * (Mh1 + Me))) * (Me * q * aB / (2 * (Mh1 + Me)));
                 Ipe = 1 / (sqrt(Ipe1) * Ipe1);
                 Iph = 1 / (sqrt(Iph1) * Iph1);
          if ( (((E2 * E2) / (hb * hb * vTA * vTA)) - q2) > 0)
                 {
                      qz = sqrt(((E2 * E2) / (hb * hb * vTA * vTA)) - q2);
                      Bp = 8 * pi * pi * sin(Lz * qz / 2) / (qz * Lz * ( (4 * pi * pi) - (Lz * Lz * qz * qz) ));
                      defepart = defe * defe * qtot;
                      defhpart = defh * defh * qtot;
                      piezopart1 = piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 / 8) )/



(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                      /*
       piezopart1 = 2 * piezo14 * piezo14 * q2 * ( (qz * qz) + (q2 * cos(phiq)* cos(phiq) * sin(phiq) *  sin(phiq)) ) /



(16 * pi * pi * eo * eo * einf * einf * qtot * qtot * qtot * qtot * qtot);
                      */
                      /* piezopart2e = defe * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



(pi * einf * (E2 * E2 / ( hb * hb * vTA * vTA)) );
                      piezopart2h = defh * piezo14 * q * (qz * (cos(phiq) + sin(phiq)) + (q * cos(phiq) * sin(phiq))) /



(pi * einf * (E2 * E2 / ( hb * hb * vTA * vTA)) ); */
                      piezopart2e = 0;
                      piezopart2h = 0;
                      Allpartse = (defepart + piezopart1 + piezopart2e) * Ipe * Ipe;
                      Allpartsh = (defhpart + piezopart1 + piezopart2h) * Iph * Iph;
                      Allpartseh = 2 * (-sqrt(defepart * defhpart) + piezopart1) * Ipe * Iph;
                 /****  W's assume area of cavity is 1 cm^2 *****/

                      NinkTA[j][jj] +=  pi * Bp * Bp * E2 * Xjj * Xj * (Allpartse + Allpartsh + Allpartseh) / (p * qz);
                 }
                 else
```

```
                    NinkTA[j][jj] += 0;
    }
}
/*if(jj > 0)
{*/
        if(j == 0)
{
             Ipe10 = 1 + (Mh1 * kpp[jj] * aB / (2 * (Mh1 + Me))) * (Mh1 * kpp[jj] * aB / (2 * (Mh1 + Me)));
             Iph10 = 1 + (Me * kpp[jj] * aB / (2 * (Mh1 + Me))) * (Me * kpp[jj] * aB / (2 * (Mh1 + Me)));
             Ipe0 = 1 / (sqrt(Ipe10) * Ipe10);
             Iph0 = 1 / (sqrt(Iph10) * Iph10);
             if( (((f[0][0] - E1) * (f[0][0] - E1) / (hb * hb * v * v)) - (kpp[jj]*kpp[jj])) > 0 )
    {
                  qz = sqrt( ((f[0][0] - E1) * (f[0][0] - E1) / (hb * hb * vTA * vTA)) - (kpp[jj]*kpp[jj]) );
                  Bp = 8 * pi * pi * sin(Lz * qz / 2) / (qz * Lz * ( (4 * pi * pi) - (Lz * Lz * qz * qz)));
                  Wden0TA[jj] = Wden1TA * qz / (2 * pi);   /* when k == 0 */
                  W0TA[jj] = Bp * Bp * (E1 - f[0][0]) * (E1 - f[0][0]) * 0.5 * Xjj *



(defe * Ipe0 - defh * Iph0) * (defe * Ipe0 - defh * Iph0);



/*Bp * Bp * hb * v * kpp[jj] * (E1 - f[0][0]) * 0.5 * Xjj * (defe * Ipe0 - defh * Iph0) * (defe * Ipe0 - defh * Iph0);*/
                  Wden00TA[jj] = Wden1TA * qz * D2A / (8 * pi * pi);    /* when k != 0 */
      }
      else
      {
           W0TA[jj] = 0;
           Wden0TA[jj] = 1;
   Wden00TA[jj] = 1;
      }
         }
/*}
else
{
W0[0] = 0;
Wden0[0] = 1;
Wden00[0] = 1;
}*/

/*          printf("W = %e \n", W[j][jj][i]); */
    }
}

for (j = 0; j <= p; j += (p-2)/2)
{
    for (i = 1; i <= p; i++)
        WtTA[j] += WTA[j][i][0] * delk[i] * kpp[i] * 2 * pi;
}

for (i = 0; i <= p; i++)
{
    for(ii = 0; ii <= p; ii++)
NinkTA[i][ii] *= 4 / Wden1TA;
}

/* ---- Used to check symmetry of the Nink ---- */

for(i = 0; i <= p; i++)
{
for(ii = 0; ii <= p; ii++)
{
            if(fabs(NinkTA[i][ii] - NinkTA[ii][i]) / fabs(NinkTA[i][ii]) > 1e-14)
        printf("Matrix element assymmetry for TA phonons, %e   %e   %e\n", NinkTA[i][ii], NinkTA[ii][i],fabs(NinkTA[i][ii] - NinkTA[ii][i]) /



fabs(NinkTA[i][ii]));
}
}
NinkTA[p][p+1] = 0;
NinkTA[p+1][p] = 0;
/* scattering matrix element vs |q|
sprintf(outfile4, "Mvsq.d", xx);
fq = fopen(outfile4, "w");
fprintf(fq, "x   ");
for(i = 0 ; i <= p; i++)
     fprintf(fq, "%e   ", kpp[i]);
```

```
fprintf(fq, "\n");
for ( i = 0; i <= p; i++)
{
fprintf(fq, "%e   ", kpp[i]);
for(ii = 0; ii <= p; ii++)
fprintf(fq, "%e   ", Nink[i][ii]);
fprintf(fq, "\n");
}
fclose(fq);
exit(1);*/

return(1);
}
```

B.20 2DPOLSCAT.H

```
int D2polscat(double f[1000][6], double Ninpol[1000], double Noutpol[1000], int m,
double del[1000], int zzz, int Geo, double kpp[1000], double delk[1000], double index, int disp, int y)
{
double Fin0, Fout0, Fin00, Fout00, dEdk3, den00, rado, dEdki2, dEdki3, kxk;

nnn=0;
/* include a loop for each integration variable in this list */
/* E, E1, E2 */
Allconst = 1.2 * 36 * EB * EB * aB * aB * aB * aB / (hb * 32 * pi * pi * pi);
den00 = 0;
if (y == 1 || disp == 3)
{
     for(i = 0; i <= p; i++)
     {
          if(i == 0)
              kxk = 1;
          else
              kxk = kpp[i] * kpp[i];
          for(ii = 0; ii <= p; ii++)
          {
               Vp[i][ii] = 0;
               if(i == 0 && ii != 0)
                   kxk1 = kpp[ii];
               else if(i != 0 && ii == 0)
                   kxk1 = kpp[i];
               else if(i == 0 && ii == 0)
                   kxk1 = 1;
               else
                   kxk1 = kpp[i] * kpp[ii];
               dEdki2 = dEdki[ii];   /* 9/19/07 change */ /* getdEdk(f[ii][0], kpp[ii], disp, index, 0, y); */
               for(iii = 0; iii <= p; iii++)
               {
                    E3 = f[iii][0] + f[ii][0] - f[i][0];
                    dEdki3 = dEdki[iii]; /* 9/19/07 change */ /*getdEdk(f[iii][0], kpp[iii], disp, index, 0, y); */
                    if (E3 > f[0][0] && E3 < f[p+1][0])
                    {
                         k3 = Determinek3(E3, f, 0);
                         I[i][ii][iii] = polD[0];
                         k2k3 = kpp[iii] * k3;
 dEdk3 = getdEdk(E3, k3, disp, index, 0, y);
                         fnew[i][ii][iii] = (k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]];



/*f[I[i][ii][iii]][1] + ( (f[1 + I[i][ii][iii]][1] - f[I[i][ii][iii]][1]) * ((k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]]) ); */
                         /*
 if(I[i][ii][iii] > 0)
     fnew[i][ii][iii] = findf(I[i][ii][iii], f, delk, (k3 - kpp[I[i][ii][iii]]) / delk[I[i][ii][iii]]);
 else
       fnew[i][ii][iiii] = f[0][1] + ((f[1][1] - f[0][1]) * (k3 / delk[0]));
                         */
                         if (i == 0)
 {
     E40[ii][iii] = f[iii][0] + f[ii][0] - f[0][0];
     k40[ii][iii] = Determinek3(E40[ii][iii], f, zzz);
     if( (k40[ii][iii] < (kpp[ii] + kpp[iii])) && ((k40[ii][iii]*k40[ii][iii]) > (kpp[ii]-kpp[iii])*(kpp[ii]-kpp[iii])) )
     {
```

```
          dEdk40[ii][iii] = getdEdk(E40[ii][iii], k40[ii][iii], disp, index, zzz, y);
  dfac0[ii][iii] = (0.5 * Xp[ii] * Xp[iii] * Xp[I[i][ii][iii]]) * deldel[ii][iii] * 2 * pi /



(dEdki2 * dEdki3 * dEdk40[ii][iii] * (sqrt((2 * kpp[iii] * kpp[iii] * kpp[ii] * kpp[ii]) - (k40[ii][iii]*k40[ii][iii]*k40[ii][iii]*k40[ii][iii]) +



(2*k40[ii][iii]*k40[ii][iii]*(kpp[ii]*kpp[ii] + kpp[iii]*kpp[iii])) - (kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) -



(kpp[iii]*kpp[iii]*kpp[iii]*kpp[iii])) ) );
                              }
     else dfac0[ii][iii] = 0;
 }
 if (iii == 0)
 {
      E400[i][ii] = f[i][0] + f[ii][0] - f[0][0];
      k400[i][ii] = Determinek3(E400[i][ii], f, zzz);
      if( (k400[i][ii] < (kpp[i] + kpp[ii])) && ((k400[i][ii]*k400[i][ii]) > (kpp[i]-kpp[ii])*(kpp[i]-kpp[ii])) )
      {
           dEdk400[i][ii] = getdEdk(E400[i][ii], k400[i][ii], disp, index, zzz, y);
           den00 = (2 * kpp[i] * kpp[i] * kpp[ii] * kpp[ii]) - (k400[i][ii] * k400[i][ii]



* k400[i][ii] * k400[i][ii]) + (2 * k400[i][ii] * k400[i][ii] * (kpp[ii]*kpp[ii] + kpp[i]*kpp[i])) -



(kpp[ii]*kpp[ii]*kpp[ii]*kpp[ii]) - (kpp[i]*kpp[i]*kpp[i]*kpp[i]);
           dfac00[i][ii] = 2 * (Xp[ii] * 0.5 * Xp[I[i][ii][iii]]) * del[ii] * 8 * pi * pi / (D2A * dEdki2 * dEdk400[i][ii] * sqrt(den00) );
                            }
      else dfac00[i][ii] = 0;
 }
 radp = (kxk - kpp[iii] * kpp[iii] - k2k3) / kxk1;
                         radn = (kxk - kpp[iii] * kpp[iii] + k2k3) / kxk1;
                         if(radp < -1)
                              thetamax = pi;
                         else if(radp > 1)
                              thetamax = 0;
                         else
                              thetamax = acos(radp);
                         if (radn > 1)
                              thetamin = 0;
                         else if (radn < -1)
                              thetamin = pi;
                         else
                              thetamin = acos(radn);
                         if (thetamax - thetamin > 1e-3) /* 1e-13 */
                         {
                             dif = thetamax - thetamin;
                             ttheta = thetamax + thetamin;
                             GQ1 = 0;
                             for(j = 1; j <= GQp; j++)
                             {
                                  rado = cos(((dif * yy[j]) + ttheta) / 2);
                                  GQ1 += ww[j] / sqrt( (radp - rado) * (-radn + rado) );
                             }
                             GQ1 *= dif / (2 * kxk1);
                             dfac1[i][ii][iii] = (Xp[ii] * Xp[iii] * Xp[I[i][ii][iii]]) * GQ1 * deldel[ii][iii] / (dEdki2 * dEdki3 * dEdk3);
                         }
                         else
                             dfac1[i][ii][iii] = 0;
                    }
                    else
                    {
                          dfac1[i][ii][iii] = 0;
                          if(i == 0) dfac0[ii][iii] = 0;
                    }
                    if (dfac0[ii][iii] != dfac0[ii][iii] || dfac0[ii][iii] > 1e50 || dfac00[i][ii] != dfac00[i][ii] ||



dfac00[i][ii] > 1e50 || dfac0[ii][iii] < -1e50 || dfac00[i][ii] < -1e50 ||
```

```
dfac1[i][ii][iii] < - 1e50 || dfac1[i][ii][iii] > 1e50 || dfac1[i][ii][iii] != dfac1[i][ii][iii])
                    {
                        printf("error in 2Dpolscat.h\n");
                        printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                        printf("df1 =  %e     df0 = %e     df00 = %e    \n", dfac1[i][ii][iii], dfac0[ii][iii], dfac00[i][ii]);
                printf("GQ = %e, dEdk2 =%e, dEdk3 = %e, dEdk4 = %e, dif = %e\n", GQ1, dEdki2, dEdki3, dEdk3, dif);
                        printf("E400 =  %e     k400 = %e     den00 = %e    \n", E400[i][ii], k400[i][ii], den00);
                        printf("dEdk400 = %e, kxk1 = %e\n", dEdk400[i][ii], kxk1);
                        exit(1);
                    }
    if(den00 < 0)
    {
         printf("den00 error in 2Dpolscat.h, k = %e, k1 = %e,k400 = %e, den00 = %e\n", kpp[i], kpp[ii], k400[i][ii], den00);
 printf("i = %d, ii = %d\n", i, ii);
                         exit(1);
    }
              }
           }
     }
}
Ninpol[p+3] = 0;
Noutpol[p+3] = 0;
                        /*     LOOP E        */
for (i = 0; i <= p; i++)
{
     Ninpol[i] = 0;
     Noutpol[i] = 0;


                        /*     LOOP E1       */
     for (ii = 0; ii <= p; ii++)
     {
                        /*     LOOP E2       */

          for (iii = 0; iii <= p; iii++)
          {
               f5 = f[I[i][ii][iii]][1] + ((f[I[i][ii][iii]+1][1] - f[I[i][ii][iii]][1]) * fnew[i][ii][iii]);
               Fin = f[ii][1] * f[iii][1] * ( 1 + f5 )  * (1 + f[i][1]);
               Fout = f[i][1] * f5 * (1 + f[ii][1]) * (1 + f[iii][1]);
               /* uncomment the following two lines to delete Bose effects
               Fin = f[ii][1] * f[iii][1];
               Fout = f5 * f[i][1];*/
             /* printf("Fin = %e Fout = %e\n", Fin, Fout); */
               if(i == 0)
               {
                    Fin0 = f[ii][1] * f[iii][1] * (1 + f5) * (1 + f[p+3][1]);
                    Fout0 = f[p+3][1] * f5 * (1 + f[ii][1]) * (1 + f[iii][1]);
               }
               if(iii == 0)
               {
                    Fin00 = f[ii][1] * f[p+3][1] * (1 + f5) * (1 + f[i][1]);
                    Fout00 = f[i][1] * f5 * (1 + f[ii][1]) * (1 + f[p+3][1]);
               }
               dNin =  dfac1[i][ii][iii] * Fin;
               if (dNin != dNin || dNin > 1e100 || dNin < -1e100)
               {
                   printf("df1 = %e, Fin = %e\n", dfac1[i][ii][iii], Fin);
                   printf("f1 = %e, f2 = %e, f3 = %e, f4 = %e\n", f[i][1], f[ii][1], f[iii][1], fnew[i][ii][iii]);
                   printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                   exit(1);
               }
               dNout = dfac1[i][ii][iii] * Fout;
               Ninpol[i] += dNin;
               Noutpol[i] += dNout;
               if(i == 0)
               {
                    Ninpol[p+3] += dfac0[ii][iii] * Fin0;
                    Noutpol[p+3] += dfac0[ii][iii] * Fout0;
               }
               if(iii == 0)
               {
                    Ninpol[i] += Fin00 * dfac00[i][ii];
                    Noutpol[i] +=  Fout00 * dfac00[i][ii];
               }
          }                         /* loop E2 */
     }                              /* loop E1 */
Ninpol[i] *= Allconst * Xp[i] * f[i][2];
Noutpol[i] *= Allconst * Xp[i] * f[i][2];
```

```
if(i == 0)
{
     Ninpol[p+3] *= DOS[p+3];
     Noutpol[p+3] *= DOS[p+3];
}
}                                   /* loop E */
return(1);
}
```

B.21 2DPOLELSCAT.H

```
int D2polelscat(double f[1000][6], double Ninpolel[1000], double Noutpolel[1000], int m,
double del[1000], int zzz, int Geo, double kpp[1000], double delk[1000], double index, int disp, int y)
{
double Fin0, Fout0, Fin00, Fout00, dEdk3, den00, rado, dEdki2, dEdki3, kxk;

nnn=0;
/* include a loop for each integration variable in this list */
/* E, E1, E2 */
if (y == 1 || disp == 3) polelMatx(f, Ninpolel, Noutpolel, m, del, zzz, Geo, kpp, delk, index, disp, y);

                        /*     LOOP E        */
for (i = 0; i <= p; i++)
{
     Ninpolel[i] = 0;
     Noutpolel[i] = 0;

                        /*     LOOP E1       */
     for (ii = 0; ii <= p; ii++)
     {
                        /*     LOOP E2       */

          for (iii = 0; iii <= p; iii++)
          {
               f5 = fe[I[i][ii][iii]] + ((fe[I[i][ii][iii]+1] - fe[I[i][ii][iii]]) * fnew[i][ii][iii]);
               Fin = f[ii][1] * fe[iii] * (1 + f[i][1]) * (1 - f5);  /* took out 1 - f5, 02/13/08*/
               Fout = f[i][1] * f5 * (1 + f[ii][1]) * (1 - fe[iii]);    /* took out 1 - fe[iii], 02/13/08 */
               if(i == 0)
               {
                    Fin0 = f[ii][1] * fe[iii] * (1 - f5) * (1 + f[p+3][1]);
                    Fout0 = f[p+3][1] * f5 * (1 + f[ii][1]) * (1 - fe[iii]);
               }
               if(iii == 0)
               {
                    Fin00 = f[ii][1] * fe[p+3] * (1 - f5) * (1 + f[i][1]);
                    Fout00 = f[i][1] * f5 * (1 + f[ii][1]) * fe[p+3];
               }
               dNin =  dfacel1[i][ii][iii] * Fin;
               if (dNin != dNin || dNin > 1e100 || dNin < -1e100)
               {
                   printf("Error in 2Dpolelscat.h, #1.\n");
                   printf("df1 = %e, Fin = %e\n", dfacel1[i][ii][iii], Fin);
                   printf("f1 = %e, f2 = %e, f3 = %e, f4 = %e\n", f[i][1], f[ii][1], f[iii][1], fnew[i][ii][iii]);
                   printf("i = %d, ii = %d, iii = %d, I = %d\n", i, ii, iii, I[i][ii][iii]);
                   exit(1);
               }
               dNout = dfacel1[i][ii][iii] * Fout;
               Ninpolel[i] += dNin;
               Noutpolel[i] += dNout;
               if (Ninpolel[i] < 0 || Noutpolel[i] < 0)
               {
                    printf("error in 2Dpolelscat.h, #2\n");
                    printf("dfac %d = %e\n", i, dfacel1[i][ii][iii]);
                    printf("Fin = %e,  Fout = %e\n", Fin, Fout);
                    exit(1);
               }
               if(i == 0)
               {
                    Ninpolel[p+3] += dfacel0[ii][iii] * Fin0;
                    Noutpolel[p+3] += dfacel0[ii][iii] * Fout0;
               }
```

```
               if(iii == 0)
               {
                    Ninpolel[i] += Fin00 * dfacel00[i][ii];
                    Noutpolel[i] +=  Fout00 * dfacel00[i][ii];
               }
          }                          /* loop E2 */
     }                               /* loop E1 */
     Ninpolel[i] *= Xp[i] * f[i][2];
     Noutpolel[i] *= Xp[i] * f[i][2];
     if(i == 0)
     {
          Ninpolel[p+3] *= DOS[p+3];
          Noutpolel[p+3] *= DOS[p+3];
     }
}                                    /* loop E */
return(1);
}
```

B.22 2DPOLPLASCAT.H

```
void D2polpscat(double f[1000][6], double Ninph[1000], double Noutph[1000], int m, double del[1000],



int zzz, int Geo, double kpp[1000], double delk[1000], int y, double index, double kcz, int disp)
{
double GQ1in, GQ1out;
double Fq0, Fq00, GQ1in0, GQ1in00, GQ1out0, GQ1out00, Ninph0, Noutph0, r, r1;
double f0;

if (y == 1 || disp == 3)
     M2(W, kpp, defpote, defpoth, index, kcz, W0, Wden0, Wden00);   /* initialize the scattering rate array */

/* Dp = 4 * pi / (hb * hb * hb * v * v * v); */
Ninph[p+3] = 0;
Noutph[p+3] = 0;
                  /*     LOOP E        */
for(i = 0; i <= p; i++)
{
     r = 0;
     k = kpp[i];
     E = f[i][0];
     Ninph[i] = 0;
     Noutph[i] = 0;
     Ninph0 = 0;
     Noutph0 = 0;
     f0 = f[i][1];

/*     dEdk0 = getdEdk(E, k); */  /* use if integrating over energy */

                   /*     LOOP E1   */
     for(ii = 0; ii <= p; ii++)
     {
          r1 = 0;   /* made random again 02/12/08 */
          k1 = kpp[ii];
          E1 = f[ii][0];
  dEdk1 = dEdki[ii];   /* getdEdk(E1, k1, disp, index, 0, y); */
  /* if(del[i] > (0.01 * kb * T)) 9/19/07 change */
          /*f0 = f[i][1]; */  /* findf(i, f, delk,  r);*/ /*f[i][1] + ((f[i+1][1] - f[i][1]) * r);*/



/*Determinef4(E, k, f, m, i, zzz, Geo, kpp, delk, del);*/
          /* else */
  /*f0 = f[i][1] + ((f[i+1][1] - f[i][1]) * r); */
  /* if(del[ii] > (0.01 * kb * T)) 9/19/07 change */
  f1 = f[ii][1];  /*findf(ii, f, delk, r1);*/  /* Determinef4(E1, k1, f, m, ii, zzz, Geo, kpp, delk, del); */
          /*else */
  /*f1 = f[ii][1] + ((f[ii+1][1] - f[ii][1]) * r1); */   /*  9/19/07 change */
  /* if(f0 < 0) f0 = (f[i-1][1] + f[i][1] + f[i+1][1] + f[i+2][1]) / 4;
  if(f1 < 0) f1 = (f[ii-1][1] + f[ii][1] + f[ii+1][1] + f[ii+2][1]) / 4; 9/19/07 change */
      if (E != E1)
```

```
            {
                 Fq = 1 / (exp(fabs(E - E1) / (kb * T)) - 1);
   if (E > E1)
   {
                     GQ1in = (1 + f0) * f1 * Fq; /* replace this 6/3/07 */
                     GQ1out = f0 * (1 + f1) * (1 + Fq); /* and this */
   }
                 else
   {
                     GQ1in = (1 + f0) * f1 * (1 + Fq); /* and this */
                     GQ1out = f0 * (1 + f1) * Fq; /* and this 6/3/07 */
   }
   dfac = (Nink[i][ii] + ((Nink[i+1][ii+1] - Nink[i][ii]) * sqrt(r * r + r1 * r1) / sqrt(2))) * kpp[ii] * delk[ii] * 2;
                 Ninph[i] += dfac * GQ1in;
                 Noutph[i] += dfac * GQ1out;
      /*  printf("dfac = %e, GQ1in = %e   GQ1out = %e\n", dfac, GQ1in, GQ1out);  */
      }
        /* scattering rate in vs |q|*/ /* uncomment fclose and exit below */
/* if(i == 0 && ii == 0)
{
sprintf(outfile4, "Rvsq.d");
fu = fopen(outfile4, "w");
fprintf(fu, "x   ");
for(j = 0 ;j <= p; j++)
fprintf(fu, "%e   ", kpp[j]);
fprintf(fu, "\n");
}

if(ii == 0)
fprintf(fu, "%e   ", kpp[i]);

       if(E == E1 || dfac != dfac || GQ1out != GQ1out)
fprintf(fu,"0    ");
else
fprintf(fu, "%e   ", dfac * GQ1out * f[i][2]);
if(ii == p)
fprintf(fu, "\n");*/

  if(i == 0 && ii != 0)
              Fq0 = 1 / (exp((E1 - f[0][0]) / (kb * T)) - 1);
  else
       Fq0 = 2 * kb * T / del[0];

          if(ii == 0 && i != 0)
               Fq00 = 1 / (exp((E - f[0][0]) / (kb * T)) - 1);
  else
       Fq00 = 2 * kb * T / del[0];
  if(i == 0)
  {
               GQ1out0 = f[p+3][1] * (1 + f1) * Fq0; /* and this */
               GQ1in0 = (1 + f[p+3][1]) * f1 * (1 + Fq0); /* and this 6/3/07 */
       Ninph[p+3] +=  W0[ii] * del[ii] * GQ1in0 / (dEdk1 * Wden0[ii]);
               Noutph[p+3] +=  W0[ii] * del[ii] * GQ1out0 / (dEdk1 * Wden0[ii]);
  }
          if(ii == 0)
  {
               GQ1in00 = f[p+3][1] * (1 + f0) * Fq00; /* and this 6/3/07 */
               GQ1out00 = (1 + f[p+3][1]) * f0 * (1 + Fq00); /* and this */
       Ninph0 =  W0[i] * GQ1in00 / Wden00[i];
               Noutph0 = W0[i] * GQ1out00 / Wden00[i];
  }

          if (Ninph[i] > 1E100 || Noutph[i] > 1E100 || Ninph[i] != Ninph[i])
  {
               printf("error in 2Dpolpscat.h\n");
               printf("GQ1in = %e, GQ1out = %e, f0 = %e, f1 = %e\n", GQ1in, GQ1out, f0, f1);
               printf("Fq = %e, E - E1 = %e\n", Fq, E - E1);
               exit(1);
  }
     /*  first E1 if statement */
/*  if((i == 0 || i == 1) && ii == 0)
  printf("Ninph %e Noutph %e Ninph0 %e Noutph0 %e\n", Ninph[i], Noutph[i], Ninph0, Noutph0);*/
  if (Noutph[i] < 0)
  {
   printf("GQ1 out %e  k %e  dEdk1 %e dfac %e\n", GQ1out, k, dEdk1, dfac);
   printf("i %d     ii %d\n", i, ii);
   printf("Ninks:  %e   %e  %e  \n", Nink[i][ii], Nink[i+1][ii+1], sqrt(r * r + r1 * r1));
   printf("f0 = %e   f1 = %e\n", f0, f1);
   printf("f(i) = %e  f(i+1) = %e  f(i+2) = %e\n", f[i][1], f[i+1][1], f[i+2][1]);
   printf("GQ1in = %e  f0 = %e  f1 = %e  Fq = %e\n", GQ1in, f0, f1, Fq);
```

```
   printf("f(i) = %e  f(i+1) = %e  f(i+2) = %e  r = %e\n", f[30][1], f[31][1], f[32][1], r);
   printf("E = %e  f(E)= %e\n", E, 1 /(exp((E - mu - f[0][0])/(kb * T)) - 1));
   printf("E1 = %e f(E1) = %e\n", E1, 1/(exp((E1 - mu - f[0][0])/(kb * T)) - 1));
   printf("E(i) = %e  E(i+1) = %e\n", f[30][0], f[31][0]);
       exit(1);
  }
      }   /*  LOOP E1  */
/* if(i == 1) exit(1);*/

/*
printf("E = %f Eq1 = %f   Eq2 = %f\n", E/kb/T, Eq1/kb/T, Eq2/kb/T);
printf("N41 = %1.12f N42 = %1.12f N5 = %1.12f N61 = %1.12f N62 = %1.12f\n", N41, N42, N5, N61, N62);
printf("qGQ1 = %f\n", qGQ1);
printf("qmax1 = %f  qmin1 = %f\n", qmax1, qmin1);
printf("GQ1in = %f\n", GQ1in);
printf("qdif1 = %f  qdif2 = %f  phcoef = %f\n", qdif1, qdif2, phcoef);
printf("Gqtotalin1 = %f   GQtotalin2 = %f\n", GQtotalin1, GQtotalin2);
printf("i = %d in = %f  out = %f\n\n", i, Ninph[i], Noutph[i]);
*/
/*printf("Nin0 = %e Nout0 = %e Nin = %e Nout = %e\n", Ninph0, Noutph0, Ninph[i], Noutph[i]);*/
Ninph[i] += Ninph0;
Noutph[i] += Noutph0;
Ninph[i] *= f[i][2];
Noutph[i] *= f[i][2];
/*
if(initial == 2 && ( (fabs(Ninph[i] - Noutph[i])/ Ninph[i]) > 4e-15) )
{

     printf("%d   LA phonon difference = %e\n", i, fabs(Ninph[i] - Noutph[i])/Ninph[i]);
     exit(1);
}
*/


/*printf("%d    %e     %e\n", i, Ninph[i], Noutph[i]);*/
}        /* LOOP E */
/*exit(1);*/
Ninph[p+3] *= DOS[p+3];
Noutph[p+3] *= DOS[p+3];
/*fclose(fu);
exit(1);*/
}
```

B.23 2DPOLPTASCAT.H

```
void D2polpTAscat(double f[1000][6], double NinphTA[1000], double NoutphTA[1000], int m, double del[1000],



int zzz, int Geo, double kpp[1000], double delk[1000], int y, double index, double kcz, int disp)
{
double GQ1in, GQ1out;
double Fq0, Fq00, GQ1in0, GQ1in00, GQ1out0, GQ1out00, Ninph0, Noutph0, r, r1;
double f0;

if (y == 1 || disp == 3)
{
     if(dopiezo == 1)
          MTA2(W, kpp, defpoteT, defpothT, index, kcz, W0TA, Wden0TA, Wden00TA);/* Calculate the scattering matriz elements  */
     else
          MTAPiezo2(W, kpp, defpoteT, defpothT, index, kcz, W0TA, Wden0TA, Wden00TA);
}
/* Dp = 4 * pi / (hb * hb * hb * v * v * v); */
NinphTA[p+3] = 0;
NoutphTA[p+3] = 0;
                /*     LOOP E        */
for(i = 0; i <= p; i++)
{
     r = 0;
     k = kpp[i];
     E = f[i][0];
     NinphTA[i] = 0;
     NoutphTA[i] = 0;
```

```
     Ninph0 = 0;
     Noutph0 = 0;

/*     dEdk0 = getdEdk(E, k); */  /* use if integrating over energy */

                  /*     LOOP E1   */
     for(ii = 0; ii <= p; ii++)
     {
          r1 = 0;
          k1 = kpp[ii];
          E1 = f[ii][0];
  dEdk1 = dEdki[ii];   /* getdEdk(E1, k1, disp, index, 0, y); */
  /* if(del[i] > (0.01 * kb * T)) 9/19/07 change */
          f0 = f[i][1];   /* findf(i, f, delk,  r);*/ /*f[i][1] + ((f[i+1][1] - f[i][1]) * r);




*/ /*Determinef4(E, k, f, m, i, zzz, Geo, kpp, delk, del);*/
          /* else */
  /*f0 = f[i][1] + ((f[i+1][1] - f[i][1]) * r); */
  /* if(del[ii] > (0.01 * kb * T)) 9/19/07 change */
  f1 = f[ii][1];  /*findf(ii, f, delk, r1);*/  /* Determinef4(E1, k1, f, m, ii, zzz, Geo, kpp, delk, del); */
          /*else */
  /*f1 = f[ii][1] + ((f[ii+1][1] - f[ii][1]) * r1);   */  /*  9/19/07 change */
  /* if(f0 < 0) f0 = (f[i-1][1] + f[i][1] + f[i+1][1] + f[i+2][1]) / 4;
  if(f1 < 0) f1 = (f[ii-1][1] + f[ii][1] + f[ii+1][1] + f[ii+2][1]) / 4; 9/19/07 change */
      if (E != E1)
             {
                  Fq = 1 / (exp(fabs(E - E1) / (kb * T)) - 1);
   if (E > E1)
   {
                      GQ1in = (1 + f0) * f1 * Fq;
                      GQ1out = f0 * (1 + f1) * (1 + Fq);
   }
   else
                  {
                      GQ1in = (1 + f0) * f1 * (1 + Fq);
                      GQ1out = f0 * (1 + f1) * Fq;
   }
   dfac = (NinkTA[i][ii] + ((NinkTA[i+1][ii+1] - NinkTA[i][ii]) * sqrt(r * r + r1 * r1) / sqrt(2))) * kpp[ii] * delk[ii] * 2;
                  NinphTA[i] += dfac * GQ1in;
                  NoutphTA[i] += dfac * GQ1out;
      /*  printf("dfac = %e, GQ1in = %e   GQ1out = %e\n", dfac, GQ1in, GQ1out);  */
      }
        /* scattering rate in vs |q|*/ /* uncomment fclose and exit below */
/* if(i == 0 && ii == 0)
{
sprintf(outfile4, "Rvsq.d");
fu = fopen(outfile4, "w");
fprintf(fu, "x   ");
for(j = 0 ;j <= p; j++)
fprintf(fu, "%e   ", kpp[j]);
fprintf(fu, "\n");
}

if(ii == 0)
fprintf(fu, "%e   ", kpp[i]);

        if(E == E1 || dfac != dfac || GQ1out != GQ1out)
fprintf(fu,"0    ");
else
fprintf(fu, "%e   ", dfac * GQ1out * f[i][2]);
if(ii == p)
fprintf(fu, "\n");*/

  if(i == 0 && ii != 0)
               Fq0 = 1 / (exp((E1 - f[0][0]) / (kb * T)) - 1);
  else
       Fq0 = 2 * kb * T / del[0];

          if(ii == 0 && i != 0)
               Fq00 = 1 / (exp((E - f[0][0]) / (kb * T)) - 1);
  else
       Fq00 = 2 * kb * T / del[0];
  if(i == 0)
  {
               GQ1out0 = f[p+3][1] * (1 + f1) * Fq0; /* and this */
               GQ1in0 = (1 + f[p+3][1]) * f1 * (1 + Fq0); /* and this 6/3/07 */
       NinphTA[p+3] +=  W0TA[ii] * del[ii] * GQ1in0 / (dEdk1 * Wden0TA[ii]);
               NoutphTA[p+3] +=  W0TA[ii] * del[ii] * GQ1out0 / (dEdk1 * Wden0TA[ii]);
```

```
  }
         if(ii == 0)
  {
              GQ1in00 = f[p+3][1] * (1 + f0) * Fq00; /* and this 6/3/07 */
              GQ1out00 = (1 + f[p+3][1]) * f0 * (1 + Fq00); /* and this */
       Ninph0 =  W0TA[i] * GQ1in00 / Wden00TA[i];
              Noutph0 = W0TA[i] * GQ1out00 / Wden00TA[i];
  }

         if (NinphTA[i] > 1E100 || NoutphTA[i] > 1E100 || NinphTA[i] != NinphTA[i])
  {
              printf("error in 2Dpolpscat.h\n");
              printf("GQ1in = %e, GQ1out = %e, f0 = %e, f1 = %e\n", GQ1in, GQ1out, f0, f1);
              printf("Fq = %e, E - E1 = %e\n", Fq, E - E1);
              exit(1);
  }
     /*  first E1 if statement */
/*  if((i == 0 || i == 1) && ii == 0)
  printf("Ninph %e Noutph %e Ninph0 %e Noutph0 %e\n", Ninph[i], Noutph[i], Ninph0, Noutph0);*/
  if (NoutphTA[i] < 0)
  {
   printf("GQ1 out %e  k %e  dEdk1 %e dfac %e\n", GQ1out, k, dEdk1, dfac);
   printf("i %d     ii %d\n", i, ii);
   printf("Ninks:  %e   %e  %e  \n", NinkTA[i][ii], NinkTA[i+1][ii+1], sqrt(r * r + r1 * r1));
   printf("f0 = %e   f1 = %e\n", f0, f1);
   printf("f(i) = %e  f(i+1) = %e  f(i+2) = %e\n", f[i][1], f[i+1][1], f[i+2][1]);
   printf("GQ1in = %e  f0 = %e  f1 = %e  Fq = %e\n", GQ1in, f0, f1, Fq);
   printf("f(i) = %e  f(i+1) = %e  f(i+2) = %e  r = %e\n", f[30][1], f[31][1], f[32][1], r);
   printf("E = %e  f(E)= %e\n", E, 1 /(exp((E - mu - f[0][0])/(kb * T)) - 1));
   printf("E1 = %e f(E1) = %e\n", E1, 1/(exp((E1 - mu - f[0][0])/(kb * T)) - 1));
   printf("E(i) = %e  E(i+1) = %e\n", f[30][0], f[31][0]);
       exit(1);
  }
      }   /*  LOOP E1  */
/* if(i == 1) exit(1);*/

/*
printf("E = %f Eq1 = %f   Eq2 = %f\n", E/kb/T, Eq1/kb/T, Eq2/kb/T);
printf("N41 = %1.12f N42 = %1.12f N5 = %1.12f N61 = %1.12f N62 = %1.12f\n", N41, N42, N5, N61, N62);
printf("qGQ1 = %f\n", qGQ1);
printf("qmax1 = %f  qmin1 = %f\n", qmax1, qmin1);
printf("GQ1in = %f\n", GQ1in);
printf("qdif1 = %f  qdif2 = %f  phcoef = %f\n", qdif1, qdif2, phcoef);
printf("Gqtotalin1 = %f   GQtotalin2 = %f\n", GQtotalin1, GQtotalin2);
printf("i = %d in = %f  out = %f\n\n", i, Ninph[i], Noutph[i]);
*/
/*printf("Nin0 = %e Nout0 = %e Nin = %e Nout = %e\n", Ninph0, Noutph0, Ninph[i], Noutph[i]);*/
NinphTA[i] += Ninph0;
NoutphTA[i] += Noutph0;
NinphTA[i] *= f[i][2];
NoutphTA[i] *= f[i][2];
/*
if(initial == 2 && ( (fabs(NinphTA[i] - NoutphTA[i]) / NinphTA[i]) > 4e-15) )
{
     printf(" %d TA phonon difference = %e \n", fabs(NinphTA[i] - NoutphTA[i]) / NinphTA[i]);
     exit(1);
}
*/
/*printf("%d    %e     %e\n", i, Ninph[i], Noutph[i]);*/
}        /* LOOP E */
/*exit(1);*/
NinphTA[p+3] *= DOS[p+3];
NoutphTA[p+3] *= DOS[p+3];
/*fclose(fu);
exit(1);*/
}
```

B.24 FVSESAVE.H

```
int fvsEsave(double del[1000], int xx, double N[1000], double f[1000][6], double Nin[1000],
```

```
double Nout[1000], double tau, double taut, char dataname[25], double Etotal,



int Norf, const char fname[12], int y, int o, double Ntotal, int Geo, double kpp[1000], double Ninpol[1000],



double Ninpolel[1000], double Ninph[1000], double Avetau[1])



 /*, double data[200], double delk[1000]) */
{

Tmid = Ntotal * pi * hb * hb * eC / (dg * (Me + Mh) * em * kb);

if( y == 1 || y == 2 || y == o )
{
sprintf(outfile1, "fvsEdata.%d.d", xx);
fq = fopen(outfile1, "w");
if (scattype == 13 || scattype == 15)
     fprintf(fq, "Energy     DOS     Number     Occ#     Nin-Nout    Nin     Nout    Ninpolel     Ninph\n");
else if (scattype == 14)
     fprintf(fq, "Energy     DOS     Number     Occ#     Nin-Nout    Nin     Nout    Ninpol     Ninph     Ninpolel\n");
else
     fprintf(fq, "Energy     DOS   Number   Occ#   Nin-Nout   Nin   Nout  Ninpol  Ninph\n");
if (scattype == 13 || scattype == 15)
     fprintf(fq, "%e    %e   %e   %e    %e    %e     %e     %e     %e\n",



f[0][0]/(kb*T), DOS[p+3], N[p+3], f[p+3][1], Nin[p+3] - Nout[p+3], Nin[p+3], Nout[p+3], Ninpolel[p+3], Ninph[p+3]);
else if (scattype == 14)
     fprintf(fq, "%e    %e   %e   %e    %e    %e     %e     %e     %e    %e\n",



f[0][0]/(kb*T), DOS[p+3], N[p+3], f[p+3][1], Nin[p+3] - Nout[p+3], Nin[p+3], Nout[p+3], Ninpol[p+3], Ninph[p+3], Ninpolel[p+3]);
else
     fprintf(fq, "%e   %e    %e   %e   %e    %e    %e     %e     %e\n",
f[0][0]/(kb * T), DOS[p+3], N[p+3], f[p+3][1], Nin[p+3] - Nout[p+3], Nin[p+3], Nout[p+3], Ninpol[p+3], Ninph[p+3]);
for (mmm = 0; mmm <= p; mmm++)
 {
      if (scattype == 13 || scattype == 15)
           fprintf(fq, "%e   %e    %e   %e   %e    %e    %e    %e     %e\n",
           f[mmm][0] / (kb * T), DOS[mmm], N[mmm]/del[mmm], f[mmm][1], Nin[mmm] - Nout[mmm], Nin[mmm], Nout[mmm], Ninpolel[mmm], Ninph[mmm]);
      else if (scattype == 14)
           fprintf(fq, "%e   %e    %e   %e   %e    %e    %e    %e     %e    %e\n",
           f[mmm][0] / (kb * T), DOS[mmm], N[mmm]/del[mmm], f[mmm][1], Nin[mmm] - Nout[mmm],
           Nin[mmm], Nout[mmm], Ninpol[mmm], Ninph[mmm], Ninpolel[mmm]);
      else
           fprintf(fq, "%e   %e    %e   %e   %e    %e    %e     %e     %e\n",
      f[mmm][0] / (kb * T), DOS[mmm], N[mmm]/del[mmm], f[mmm][1], Nin[mmm] - Nout[mmm],
      Nin[mmm], Nout[mmm], Ninpol[mmm], Ninph[mmm]);
 }

fclose(fq);
}


if (xx == 0)
{
     sprintf(outfile2, fname);
     fr = fopen(outfile2, "w");
     fprintf(fr, fname);
 fprintf(fr, "\nNtotal = %e\n", Ntotal);
     fprintf(fr, "\nave E = %f", Etotal / kb / T / Ntotal);
     fprintf(fr, "\np = %d\n", p);
     fprintf(fr, "uprate = %f\n", uprate[0]);
     fprintf(fr, "GQp = %d\n", GQp);
     fprintf(fr, "Geo = %d\n", Geo);
     fprintf(fr, "initial = %d\n", initial);
     fprintf(fr, "qo = %e     Ne = %e\n", qo[0], Ne[0]);
     fprintf(fr, "count = %d\n", count);
     fprintf(fr, "delc = %d\n", delc);
```

```
    fprintf(fr, "disp = %d\n", disp);
    fprintf(fr, "iterations = %d\n", o);
    fprintf(fr, "Norf = %d\n", Norf);
    fprintf(fr, "GQp = %d\n", GQp);
    fprintf(fr, "Geo = %d\n", Geo);
    fprintf(fr, "scattype = %d\n", scattype);
    fprintf(fr, "statype = %d\n", statype);

    fprintf(fr, "T = %f, TTT = %f\n", T, TTT[0]);
    fprintf(fr, "%d    ", 0);

    for ( mmm = 0; mmm <= p; mmm++)
         fprintf(fr, "%e    ", del[mmm]);
    fprintf(fr, "\n");
    fprintf(fr, "0    ");   /* energy points on mesh */
    for ( mmm = 0; mmm <= p; mmm++)
         fprintf(fr, "%e    ", (f[mmm][0] - f[0][0])*1000);    /* displays Energy in meV */
    fprintf(fr, "tau     ");
    fprintf(fr, "Etotal/kb/T     ");
    fprintf(fr, "Stime[0]     ");
    fprintf(fr, "Stime[(p-2)/2]     ");
    fprintf(fr, "Stime[p-2]     ");
fprintf(fr, "Stime[p+3]     ");
    fprintf(fr, "taut     ");
    fprintf(fr, "Tmid/log(1+f[0][1])      "); /* temperature for an equilibrium distribution, numerator related to n and nQ, 0.00045401*/
    fprintf(fr, "Nscat[1]   Nscat[p/2]    Nout[1]    Nout[p/2]     ");
    fprintf(fr, "Nin[0]/N[0]     ");
    fprintf(fr, "Ntau     ");
    fprintf(fr, "Ntotal    ");
    fprintf(fr, "Uprate    ");
    fprintf(fr, "Avetau    ");
    if (disp == 3)
        fprintf(fr, "Every_other_line_is_Renormalized_energies");
    fprintf(fr, "\n");
    if ( Geo == 5)
         fprintf(fr,"%e    ", kpp[0]);
    else
         fprintf(fr,"%e    ", (f[0][0]/(kb * T)));
    for ( mmm = 0; mmm <= p; mmm++)
    {
         if ( Geo == 5)
              fprintf(fr,"%e    ", kpp[mmm]);
         else
              fprintf(fr,"%e    ", (f[mmm][0]/(kb * T)));
    }
    fprintf(fr,"\n");
}
    if (Norf == 1)
    {
fprintf(fr, "%e    ", N[p+3]);
        for ( mmm = 0; mmm <= p+3; mmm++)
          /*   fprintf(fr, "%e    ", f[mmm][3]); */
            fprintf(fr, "%e    ", N[mmm]/del[mmm]); /* N/E */
    }
    else
    {
fprintf(fr, "%e    ", (f[p+3][1]));
        for ( mmm = 0; mmm <= p; mmm++)
       fprintf(fr, "%e    ", (f[mmm][1]));
    }
    fprintf(fr, "%1.6f   ", tau);
    fprintf(fr, "%1.6f   ", Etotal / kb / T);
    fprintf(fr, "%e     ", Stime[0]);
    fprintf(fr, "%e     ", Stime[(p-2)/2]);
    fprintf(fr, "%e     ", Stime[p-2]);
    fprintf(fr, "%e     ", Stime[p+3]);
    fprintf(fr, "%e     ", taut);
    fprintf(fr, "%e     ", Tmid / log (1 + f[0][1]) ); /* temperature for an equilibrium distribution, numerator related to n and nQ, 0.00045401*/
    fprintf(fr, "%e     %e    %e     %e     ", Nscat[1], Nscat[p/2], Nout[1], Nout[p/2]);
    fprintf(fr, "%e     ", Nin[0] / N[0]);
    fprintf(fr, "%e     ", Ntau);
    fprintf(fr, "%e     ", Ntotal);
    fprintf(fr, "%e     ", uprate[0]);
    fprintf(fr, "%e     ", Avetau[0]);
    if (disp == 3)
    {
        fprintf(fr, "\n");
        fprintf(fr, "0     ");
        for (mmm = 0; mmm <= p; mmm++)
            fprintf(fr, "%e     ", (f[mmm][0] - f[0][0]) * 1000);    /* displays energy in meV */
```

```
     }
     fprintf(fr, "\n");
/*
     if ( xx == o)
     {
      fprintf(fr, "program time (in hours) = %f\n", (float)(clock())/3600000000); */
         if ( y == o)
         {
      fprintf(fr, "%e    ", (double)(f[p+3][1]));
              for (mmm = 0; mmm <= p; mmm++)
                   fprintf(fr, "%e    ", (double)(f[mmm][1]));
              fprintf(fr, "final uprate = %e     ", uprate[0]);
              fclose(fr);
         }
return(1);
}
```

B.25 GAULEG.H

```
#include <math.h>
#define EPS 3.0e-11

void gauleg(float x1, float x2, double yy[], double ww[], int n)
{
int m,j,i;
double z1,z,xm,xl,pp,p3,p2,p1;

m=(n+1)/2;
xm=0.5*(x2+x1);
xl=0.5*(x2-x1);
for (i = 1; i <= m; i++)
        {
z=cos(3.141592654*(i-0.25)/(n+0.5));
do
                {
p1=1.0;
p2=0.0;
for (j=1;j<=n;j++)
                        {
        p3=p2;
     p2=p1;
     p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
}
pp=n*(z*p1-p2)/(z*z-1.0);
z1=z;
z=z1-p1/pp;
}
                while ( fabs(z-z1) > EPS );
yy[i]=xm-xl*z;
yy[n+1-i]=xm+xl*z;
ww[i]=2.0*xl/((1.0-z*z)*pp*pp);
ww[n+1-i]=ww[i];
}
}
#undef EPS
```

B.26 INITIATE.H

```
int initiate(int initial, double del[1000], int p, double DOS[1000], int m, double f[1000][6], double Nin[1000],



double Nout[1000], double N[1000], double mu, double hw, double expp, double Tm, int statype, double Ec, double indexr)
{
switch(initial)
```

```
{
case 0:
    fromfile(f, del, expp, indexr, 1);
    break;

case 1:
    uniformfvsE(del, p, DOS, m, f, N, expp, Tm, indexr, 1);
    break;

case 2:
    boltzmannfvsE(del, p, DOS, f, N, expp, statype, indexr, 1);
    break;

case 3:
    phononfvsE(p, b);
    gaussianfvsE(del, p, f, No, expp, Ec, indexr, 1);
    break;

case 4:
    gaussianfvsE(del, p, f, No, expp, Ec, indexr, 1);
    break;

case 5:
    exit(1);

case 6:
    neareq(del, p, DOS, f, N, expp, indexr, 1);
    break;

case 7:
pulsefvsE(del, p, DOS, f, N, expp, statype, indexr, 1);
break;

case 8:
pulsegaussianfvsE(del, p, f, No, expp, Ec, indexr, 1);
break;
case 9:
        pulseflatf(del, p, DOS, f, N, expp, indexr, 1);
        break;
}
return(1);
}
```

B.27 SCAT.H

```
int scatter(int scattype, double f[1000][6], double Nin[1000], double Nout[1000],



int m, int zzz, double del[1000], int Geo, double kpp[1000], double delk[1000], int y, double indexr, double kcz, int disp)
{
switch(scattype)
{
case 1:
      D3bosescat(f, Nin, Nout, m, Geo, kpp, delk);  /*  boson-boson scattering */
      break;

case 2:
      D2bosescat(f, Ninbos, Noutbos, m, del, zzz, Geo, kpp, delk);
  for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninbos[i];
           Nout[i] = Noutbos[i];
      }
      Nin[p+3] = Ninbos[p+3];
      Nout[p+3] = Noutbos[p+3];
      break;


case 3:
      D3xpEexchange(f, b, Nin, Nout, del, zzz, Geo, kpp, delk);
      break;
```

```
case 4:
/*    D2xpEexchange(f, b, Nin, Nout, del, zzz, Geo, kpp, delk);
      break;
*/
      D2bospscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y);
      for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninph[i];
           Nout[i] = Noutph[i];
      }
      Nin[p+3] = Ninph[p+3];
      Nout[p+3] = Noutph[p+3];
      break;
case 5:
      GQtest(f, Nin, Nout, m, del, zzz, Geo, kpp, delk);
      break;

case 6:
      if (y == 1) polfrac();
      D2polscat(f, Ninpol, Noutpol, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      for (i = 0; i <= p; i++)
      {
          Nin[i] = Ninpol[i];
          Nout[i] = Noutpol[i];
      }
      Nin[p+3] = Ninpol[p+3];
      Nout[p+3] = Noutpol[p+3];
      break;

case 7:
      if ( y == 1) polfrac();
      D2polpEexchange(f, b, Nin, Nout, del, zzz, Geo, kpp, delk, indexr, disp, y);
      break;

case 8:
      if (y == 1) polfrac();
      D2polpscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polpTAscat(f, NinphTA, NoutphTA, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      for (i = 0; i <= p; i++)
      {
          Nin[i] = Ninph[i] + (2 * NinphTA[i]);
          Nout[i] = Noutph[i] + (2 * NoutphTA[i]);
      }
      Nin[p+3] = Ninph[p+3] + NinphTA[p+3];
      Nout[p+3] = Noutph[p+3] + NoutphTA[p+3];
      break;

case 9:
      if (y == 1) polfrac();
      D2polscat(f, Ninpol, Noutpol, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      D2polpscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polpTAscat(f, NinphTA, NoutphTA, m, del, zzz, Geo, kpp, delk,y, indexr, kcz, disp);
      for (i = 0; i <= p; i++)
      {
           Nin[i] = Ninph[i] + Ninpol[i] + (2 * NinphTA[i]);
           Nout[i] = Noutph[i] + Noutpol[i] + (2 * NoutphTA[i]);
      }
      Nin[p+3] = Ninph[p+3] + Ninpol[p+3] + (2 * NinphTA[p+3]);
      Nout[p+3] = Noutph[p+3] + Noutpol[p+3] + (2 * NoutphTA[p+3]);
      break;

case 10:
      D3fermiscat(f, Nin, Nout, m, Geo, kpp, delk);  /* 3D fermi scattering */
      break;

case 11:
      if (y == 1) polfrac();
      D2polscat(f, Ninpol, Noutpol, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      D2polpscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polFscat(f, NinphF, NoutphF, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninph[i] + Ninpol[i] + NinphF[i];
           if(Nin[i] != Nin[i] || Nin[i]  > 1e50 || Nin[i] < -1e50)
           {
               printf("i = %d, Ninph = %e, Ninpol = %e, NinphF = %e\n", i, Ninph[i], Ninpol[i], NinphF[i]);
               exit(1);
           }
           Nout[i] = Noutph[i] + Noutpol[i] + NoutphF[i];
      }
```

```
      Nin[p+3] = Ninph[p+3] + Ninpol[p+3] + NinphF[p+3];
      Nout[p+3] = Noutph[p+3] + Noutpol[p+3] + NoutphF[p+3];
      break;

case 12:
      D2bosescat(f, Ninbos, Noutbos, m, del, zzz, Geo, kpp, delk);
      D2bospscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y);
      for (i = 0; i <= p; i++)
      {
           Nin[i] = Ninph[i] + Ninbos[i];
           Nout[i] = Noutph[i] + Noutbos[i];
      }
      Nin[p+3] = Ninph[p+3] + Ninbos[p+3];
      Nout[p+3] = Noutph[p+3] + Noutbos[p+3];
      break;

case 13:
      if (y == 1) polfrac();
      D2polelscat(f, Ninpolel, Noutpolel, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      D2polpscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polpTAscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninph[i] + Ninpolel[i] + (2 * NinphTA[i]);
           Nout[i] = Noutph[i] + Noutpolel[i] + (2 * NoutphTA[i]);
      }
      Nin[p+3] = Ninph[p+3] + Ninpolel[p+3] + (2 * NinphTA[p+3]);
      Nout[p+3] = Noutph[p+3] + Noutpolel[p+3] + (2 * NoutphTA[p+3]);
      break;

case 14:
      if (y == 1) polfrac();
      D2polelscat(f, Ninpolel, Noutpolel, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      D2polpscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polpTAscat(f, Ninph, Noutph, m, del, zzz, Geo, kpp, delk, y, indexr, kcz, disp);
      D2polscat(f, Ninpol, Noutpol, m, del, zzz, Geo, kpp, delk, indexr, disp, y);
      for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninpol[i] + Ninpolel[i] + Ninph[i] + (2 * NinphTA[i]);
           Nout[i] = Noutpol[i] + Noutpolel[i] + Noutph[i] + (2 * NoutphTA[i]);
      }
      Nin[p+3] = Ninpol[p+3] + Ninpolel[p+3] + Ninph[p+3] + (2 * NinphTA[p+3]);
      Nout[p+3] = Noutpol[p+3] + Noutpolel[p+3] + Noutph[p+3] + (2 * NoutphTA[p+3]);
      break;

case 15:
      if (y == 1) polfrac();
      D2polelscat(f, Ninpolel, Noutpolel, m, del, zzz, Geo, kpp, delk,indexr, disp, y);
      for(i = 0; i <= p; i++)
      {
           Nin[i] = Ninpolel[i];
           Nout[i] = Noutpolel[i];
      }
      Nin[p+3] = Ninpolel[p+3];
      Nout[p+3] = Noutpolel[p+3];
      break;

}
return(1);
}
```

APPENDIX C

CODE USER MANUAL

C.1 INTRODUCTION

The function of this code is to take a nonequilibrium distribution of particles and simulate how the particles interact with each other and a lattice. The goal is to see what the steady state solution is for a particular density and how long that steady state takes to come about in simulated time. The code has the following organization chart, Figure C.1. As can be seen on the chart the main program uses three functions to start the calculation, *variables.h*, *constants.h*, and *parameters.h*. In this Appendix, words in italics are functions and boldface words are variables. Once the program has been initialized the code simply runs through a loop that calculates the scattering based on the types of scattering the user wants, *scat.h*, updating the occupation numbers or densities based on the scattering rates and time step, *updatef.h*, and finally saving the values of the calculation at defined iteration intervals, *fvsEsave.h*.

C.2 HEADER FUNCTIONS

C.2.1 Inputs

Variables.h is a collection of global variables. These are global quantites that the calculation uses but do not need to be set to any particular value at the beginning. For example, the gaussian quadrature weights, **ww[], are calculated when the code runs. Since the code allows one to set the number of points used in the gaussian quadrature it would be impossible to set these values firmly at the beginning. Many of these variables have their definition as a comment to their right. There should be no reason to change these variables unless new code is added. The variables are listed in alphabetical order.**

***Constants.h* is a collection of physcial constants. These are definite constants like vc, the speed of light, as well as material constants like einf, the dielectric constant of a material (like GaAs). All of these quantities have their definition to their right. The physcial constants should not need to be changed unless more precision is desired. The material constants and environmental constants should be set as is appropriate. The quantities are listed in alphabetical order.**

***Parameters.h* holds constants specific to a particular run of the code. Many are flags used by the code so that it can follow correct path. Examples of these are o, the number of iterations to be done, and fname[], the name of the file to be recorded. All of these quantites have their definition to their right. Since these are the main user inputs a detailed explanation of these values will be given. The variables are listed in alphabetical order.**

counts-The number of iterations the code performs between saving the values of the calculation. The program generates an enormous amount of numerical data at each iteration. To save all of this information would quickly fill up all the available space in a file.

delc-(flag)The way the mesh is distributed. The code can set up a variety of meshes. Most of the meshes are set up so that there are more points near E=0

and fewer points at the highest energies. xxx Do the calculations of why specific formulas are used need to be presented? xxx

disp - (flag)The kind of dispersion used in the calculation.

g - The rate at which particles are added when simulating states being uniformly pumped.

Geo - (flag)The geography for the simulation.

GQp - The number of points used in the gaussian quadrature procedure.

hw - The difference in energy levels for a harmonic potential.

inj - The rate at which particles are added when simulating state being pumped with a gaussian profile.

initial - (flag)The type of pumping to be used during the simulation.

kbp - I don't believe this is currently used, however, I am reluctant to remove anything from the code.

kxy - The input wavevector if pumping polaritons at a specific k value.

maxkp - The maximum value for the wavevector. Used for polaritons. Used by *del.h*.

maxkT - The maximum value for the energy in units of kT. Used for excitons. Used by *del.h*.

maxtime - The maximum amount of simulated time that will be run. The code changes the time step based on how close to equilibrium the system is. The time steps get closer as equilibrium is approached. This value reduces the amount run time the calculation requires if equilibrium is reached.

mu - The chemical potential in units of kT.

fname[] - The name of the main file to be created. Note, there are other files created by the code called 'fvsEsave' files. These files are generally overwritten each time the code is run and have been used to monitor how well the code is working.

ynum[] - Used to monitor output of the code.

Norf - (flag)Specifies whether the code records the density of the particles or the occupation number of the states.

o - The total number of iterations the code runs through.

p - The number of points on the mesh.

pulseT - The length of simulation time that a pump pulse lasts.

qo - The screening parameter.

Sa - The area of the two dimensional system being considered.

scattype -(flag)The type of scattering that is simulated by the code. See the Scattering subsection below.

statype - (flag)The type of statistics used, Boson or Fermions.

uprate - The fraction of particles that get moved around by the scattering. At most this value can be one.

TT - Used for simulating a system of particles that are not in equilibrium with the lattice(T).

C.2.2 Initialization

The main program then takes the inputs and makes the calculations necessary to start the simulation. Primarily it sets up the mesh and provides an initial population based on the type of pumping being simulated.

***initiate.h* is the primary function called to perform this. It then calls the necessary subfunctions.**

***del.h* creates the mesh and defines del[] and f[][]. For polaritons it also calculates kpp[].**

***DOS.h* calculates the density of states for the system.**

***pulseflat.h* pumps each point on the mesh equally.**

***pulseflat.h* pumps each point on the mesh equally up to a certain cutoff. The cutoff is relative to the ground state and is set within the function itself.**

***pulsegauss.h* pumps the system with a gaussian distribution along the mesh.**

***pulseadd.h* pumps the system with an equilibrium distribution based on what stattype is.**

***fromfile.h* reads the initials occupation numbers or densities for the mesh**

from a saved file. The number of points on the mesh must be equal to the number of points in the file.

neareq.h pumps the system with a distribution that is nearly in equilbrium.

C.2.3 Scattering

After the simulation has been initialized and at the beginning of each iteration it calculates the scattering rate based on the current occupation levels. *scat.h* is the primary function that directs this part of the calculation.

2Dbosescat.h - Used for the scattering rate of two dimensional excitons with two dimensional excitons.

2Dbospscat.h - Used for the scattering rate fo two dimensional excitons with three dimensional acoustic phonons.

2Dharmscat.h - Used for the scattering of particles in a trap.

2DpolFscat.h - Used for scattering two dimensional polaritons with three dimensional optical phonons.

2DpolpEexchange.h -

3Dbosescat - Used for scattering three dimensional excitons with three dimensional excitons.

3Dfermiscat - Used for scattering three dimensional fermions with three dimensional fermions.

3DxpEexchange.h -

polpMatx.h calculates the matrix element of polaritons scattering with acoustic phonons.

polFMatx.h calculates the matrix element of polaritons scattering with optical phonons.

The calculations of the matrix elements for excitons scattering with excitons and polaritons scattering with polaritons is written into their respective scattering functions.

C.2.4 Updating

Once the scattering has been determined one function, *updatef.h*, is called to change the occupation levels at each point in the mesh. While making the changes to the occupation numbers the time step is also determined. Using this time step, pumping and lifetime effects are calculated.

C.2.5 Renormalization

Using the new occupation level *polReNorm.h* can be called. For polaritons it can be used to calculate the shift in energies due to particle-particle interaction and phase space filling.

The new energy dispersion can be degenerate. *MinEn.h* finds the mimimum of the new dispersion relationship.

C.2.6 Saving

***fvsEsave.h* is used to record the calculation's results. It has two functions. The first is to store some of the initial values of the calculation in a fvsEsave.d file. These files are used to monitor the quality of the calculation. The main file is the one put in the variable fname[]. This file stores some of the starting parameter values. After these values it stores the energy steps on the mesh, the energy points on the mesh, the wavevector equivalents on the mesh in successive lines. Then each line thereafter has the density or occupation number (depending on what Norf is set to) for each point on the mesh. At the end of each line values of the simulated time and total density are stored. Thus a grid is set up so that the values in each column fall under the mesh point to which they correspond. Each row repsresents the calculation at some iteration and the simulated time can be found near the end of the row. If disp $= 3$ then every other row is the new dispersion relationship.**

C.2.7 Miscellaneous Functions

Errors.h **contains all errors functions.**

gauleg.h **has the code for calculating the weights and points used for Gaussian Quadrature. [24]**

polfraction.h **calculates the excitonic fraction of the polaritons along the mesh.**

polpara.h **calculates the $\vec{k}_z$ constant for the calculation.**

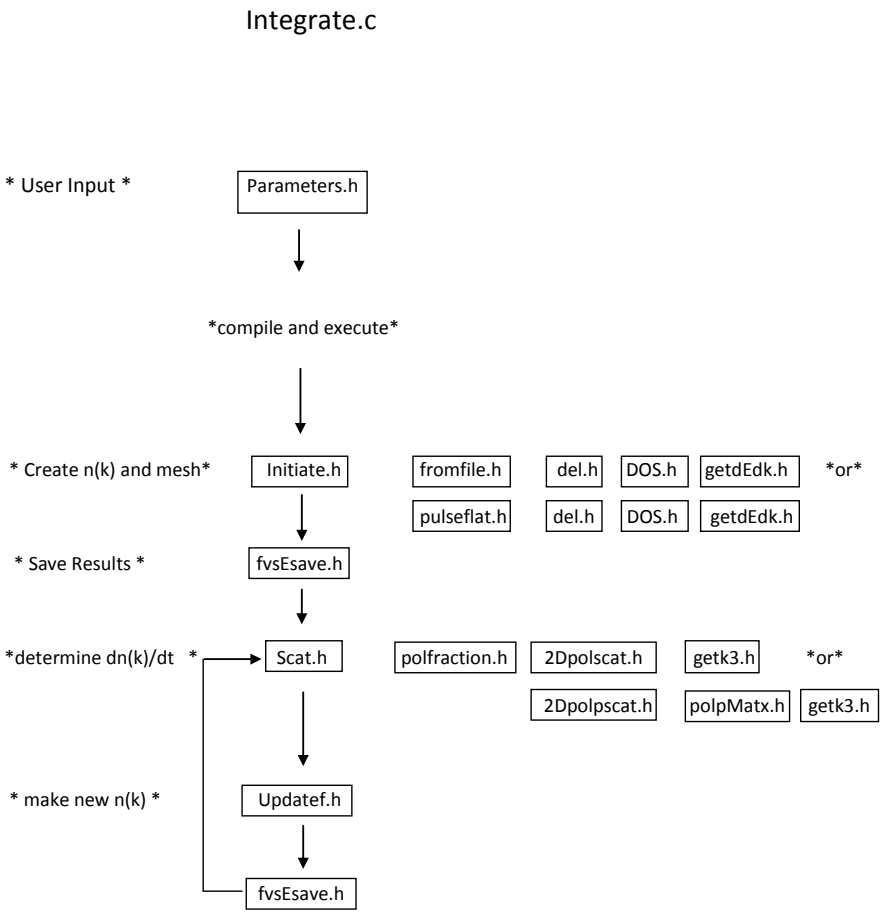

Figure C1: A flow chart of the main code

BIBLIOGRAPHY

[1] CHRISTOPOULOS, S., VON HOGERSTHAL, G. B. H., GRUNDY, A., LAGOUDAKIS, P., KAVOKIN, A., et al.,
Phys Rev Lett 98 (2007) 126405.

[2] ADACHI, S.,
J Appl Phys 58 (1985) R1.

[3] GEHRSITZ, S., REINHART, F., GOURGON, C., HERRES, N., VONLANTHEN, A., et al.,
J Appl Phys 87 (1987) 7825.

[4] BLAKEMORE, J.,
J Appl Phys 53 (1982) p. 10.

[5] JENKINS, D.,
J Appl Phys 68 (1990) 1848.

[6] SNOKE, D. W.,
Solid State Physics: Essential Concepts,
Addison-Wesley, 2008.

[7] ASPNES, D.,
Phys Rev B 14 (1976) p. 5331.

[8] ANDREANI, L. and PASQUARELLO, A.,
Phys Rev B 42 (1990) p. 8928.

[9] HOPFIELD, J.,
Phys Rev 112 (1958) 1555.

[10] KOSTERLITZ, J. and THOULESS, D.,
J Phys C 6 (1973) 1181.

[11] SAVONA, V. and SARCHI, D.,
Physica Status Solidi(b) 242 (2005) 2290.

[12] TASSONE, F., PIERMAROCCHI, C., SAVONA, V., QUATTROPANI, A., and SCHWENDIMANN, P.,
Phys Rev B 56 (1997) 7554.

[13] MULLER, M., BLEUSE, J., ANDREA, R., and ULMER-TUFFIGO, H.,
Physica B 272 (1999) 476.

[14] TARTAKOVSKII, A., EMAM-ISMAIL, M., STEVENSON, R., SKOLNIC, M., ATRATOV, V., et al.,
Phys Rev B 62 (2000) R2283.

[15] DENG, H., WEIHS, G., SNOKE, D., BLOCH, J., and YAMAMOTO, Y.,
PNAS 100 (2003) 15318.

[16] WEIHS, G., DENG, H., SNOKE, D., and YAMAMOTO, Y.,
Phys Stat Sol 201 (2004) 625.

[17] NEGOITA, V., SNOKE, D., and EBERL, K.,
App Phys Lett 14 (1999) 2059.

[18] BALILI, R., SNOKE, D., PFEIFFER, L., and WEST, K.,
App Phys Lett 88 (2006) 031110.

[19] HOHENBERG, P.,
Phys Rev 158 (1967) 383.

[20] BAGNATO, V. and KLEPPNER, D.,
Phys Rev A 44 (1991) p. 7439.

[21] BERMAN, O., LOZOVIK, Y., and SNOKE, D.,
Phys Rev B 77 (2008) 155317.

[22] DALFOVO, F., GIORGINI, S., PITAEVSKII, L., and STRINGARI, S.,
Rev Mod Phys 71 (1999) 463.

[23] BANYAI, L.,
Private Communication (2004).

[24] PRESS, W., FLANNERY, B., TEUKOLSKY, S., and VETTERLING, W.,
Numerical Recipes in C:The Art of Scientific Computing,
Press Syndicate of the University of Cambridge, 2nd edition, 1992.

[25] CIUTI, C., SAVONA, V., PIERMAROCCHI, C., QUATTROPANI, A., and SCHWENDIMANN, P.,
Phys Rev B 58 (1998) 7926.

[26] TASSONE, F. and YAMAMOTO, Y.,
Phys Rev B 59 (1999) 10830.

[27] PIERMAROCCHI, C., TASSONE, F., SAVONA, V., and QUATTROPANI, A.,
Phys Rev B 53 (1996) 15834.

[28] BASTARD, G. and BRUM, J.,
IEEE Journal of Quantum Electronics QE-22 (1986) 1625.

[29] PAU, S., BJORK, G., JACOBSON, J., CAO, H., and YAMAMOTO, Y.,
Phys Rev B 51 (1995) 7090.

[30] SNOKE, D.,
Phys Rev B 44 (1991) 2991.

[31] PIKKUS, G. and BIR, G.,
Fiz Tverd Tela 1 (1959) 1642.

[32] ADACHI, S.,
GaAs and Related Materials: Bulk Semiconducting and Superlattice Properties,
World Scientific, 1994.

[33] ARLT, G. and QUADFLIEG, P.,
Phys. Status Solidi 25 (1968) p. 323.

[34] MALPUECH, G., KAVOKIN, A., CARLO, A. D., and BAUMBERG, J.,
Phys Rev B 65 (2002) 153310.

[35] SNOKE, D.,
Phys Rev B 50 (1994) 11583.

[36] SNOKE, D. and WOLFE, J.,
Phys Rev B 39 (1989) 4030.

[37] TASSONE, F., PIERMAROCCHI, C., SAVONA, V., QUATTROPANI, A., and SCHWENDIMANN, P.,
Phys Rev B 53 (1996) R7642.

[38] MALPUECH, G., RUBO, Y., LAUSSY, F., BIGENWALD, P., and KAVOKIN, A.,
Semicond Sci Techno 18 (2003) S395.

[39] RAPAPORT, R., HAREL, R., COHEN, E., RON, A., and LINDER, E.,
Phys Rev Lett 84 (2000) 1607.

[40] GALBRAITH, I. and KOCH, S.,
J Cryst. Growth 159 (1996) 667.

[41] DOAN, T. and THOAI, B. T.,

Sol Stat Comm 123 (2002) 427.

[42] CAO, H., DOAN, T., THOAI, D., and HAUG, H.,
Phys Rev B 69 (2004) 245325.

[43] DOAN, T., CAO, H. T., THOAI, D. T., and HAUG, H.,
Phys Rev B 72 (2005) 085301.

[44] DOAN, T., CAO, H. T., THOAI, D. T., and HAUG, H.,
Phys Rev B 74 (2006) 115316.

[45] PORRAS, D., CIUTI, C., BAUMBERG, J., and TEJEDOR, C.,
Phys Rev B 66 (2002) 085304.

[46] CHAVES, F. and RODRIQUES, F.,
Sol Stat Comm 136 (2005) 484.

[47] SARCHI, D. and SAVONA, V.,
arXiv:cond-mat/0411084v3 (2006).

[48] SARCHI, D. and SAVONA, V.,
Solid State Communications 144 (2007) 371.

[49] CASTIN, Y. and DUM, R.,
Phys Rev A 57 (1998) 3008.

[50] GARDINER, C. and ZOLLER, P.,
Phys Rev A 58 (1998) 536.

[51] SERMAGE, B., LONG, S., ABRAM, I., MARZIN, J., BLOCH, J., et al.,
Phys Rev B 53 (1996) 16516.

[52] SHAH, J.,
IEEE J of Q. Elec 24 (1988) 276.

[53] WANG, X., ZHANG, G., ZHAO, Y., FAN, F., LIU, H., et al.,
Optical Materials 97 (2007) 1658.

[54] AMBACHER, O., SMART, J., SHEALY, J., WEINMANN, N., CHU, K.,
et al.,
J Appl Phys 85 (1999) 3222.

[55] HUBNER, K.,
Phys Status Solidi B 57 (1973) 627.

[56] LANGBEIN, W. and HVAM, J.,
Phys Rev Lett 8 (2002) 047401.

[57] KAVOKIN, A., BAUMBERG, J., MALPUECH, G., and LAUSSY, F.,

Microcavities,
Oxford University Press, 2007.

[58] SZYMANSKA, M., KEELING, J., and LITTLEWOOD, P.,
Phys Rev Lett 96 (2006) 230602.

[59] SARCHI, D. and SAVONA, V.,
Phys Status Solidi B 243 (2006) 2317.

[60] DOAN, T., CAO, H. T., THOAI, D. T., and HAUG, H.,
Solid State Comm 144 (2007) 359.

[61] BERMAN, O., LOZOVIK, Y., SNOKE, D., and COALSON, R.,
Solid State Communications 134 (2005) 23.

[62] FISCHER, D. and HOHENBERG, P.,
Phys Rev B 37 (1988) 4936.