Nano-scale Thermal Property Prediction by Molecular Dynamics Simulation with Experimental Validation

Kyle S. Horne
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Engineering Commons

Recommended Citation
NANO-SCALE THERMAL PROPERTY PREDICTION BY MOLECULAR DYNAMICS SIMULATION WITH EXPERIMENTAL VALIDATION

by

Kyle S. Horne

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Mechanical Engineering

Approved:

Dr. Heng Ban
Major Professor

Dr. Byard Wood
Committee Member

Dr. Robert E. Spall
Committee Member

Dr. Nick Roberts
Committee Member

Dr. Stephen Bialkowski
Committee Member

Dr. Mark R. McLellan
Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2014
Abstract

Nano-scale Thermal Property Prediction by Molecular Dynamics Simulation with Experimental Validation

by

Kyle S. Horne, Doctor of Philosophy
Utah State University, 2014

Major Professor: Dr. Heng Ban
Department: Mechanical and Aerospace Engineering

Quantum cascade laser (QCL) diodes have potential applications in many areas including emissions analysis and explosives detection, but like many solid-state devices they suffer from degraded performance at higher temperatures. To alleviate this drawback, the thermal properties of the QCL diodes must be better understood. Using molecular dynamics (MD) and photothermal radiometry (PTR), the thermal conductivity of a representative QCL diode is computed and measured respectively.

The MD results demonstrate that size effects are present in the simulated systems, but if these are accounted for by normalization to experimental results the thermal conductivity of the QCL can be reasonably obtained. The cross-plane conductivity is found to be in the range of 1.8 to 4.3 \( \text{W/m} \cdot \text{K} \), while the in-plane results are in the range of 3.7 to 4.0 \( \text{W/m} \cdot \text{K} \). These values compare well with experimental results from the literature for both QCL materials and for AlInAs and GaInAs, which the QCL is composed of. The cross-plane conductivity results are lower than those of either AlInAs or GaInAs, which demonstrates the phonon scattering at the interfaces. The in-plane results are between AlInAs and GaInAs, which is to be expected.
The PTR results are less concrete, as there seem to be heat transfer effects active in the samples which are not included in the models used to fit the frequency scans. These effects are not 2D heat transfer artifacts nor are they the result of volumetric absorption. It is possible that they are the results of plasmon induction, but this is only supposition. As the data stand, the PTR and MD results are within an order of magnitude of each other and follow reasonable trends, which suggests that both results are not too far off from reality.

While the experimental results are not entirely conclusive, the simulations and experiments corroborate each other sufficiently to warrant further investigation using these techniques. Additionally, the simulations present sufficient internal consistency so as to be useful for thermal property investigation independent of the PTR results.
Public Abstract

Nano-scale Thermal Property Prediction by Molecular Dynamics Simulation with Experimental Validation

by

Kyle S. Horne, Doctor of Philosophy
Utah State University, 2014

Major Professor: Dr. Heng Ban
Department: Mechanical and Aerospace Engineering

A new type of laser diode, called a quantum cascade laser (QCL), was developed in the mid 1990’s. These new lasers have many applications including industrial emissions analysis and explosives detection. Like many solid-state devices, they work better at cooler temperatures, but operating the device generates heat; this results in a cooling problem if the device is to operate continuously at high-power. To improve this situation, a better understanding of how heat leaves the laser diode is needed. The thermal conductivity of a material is a measure of how quickly heat will leave it. In this work, two approaches are used to better understand thermal conduction in laser diodes. First, the motions of the atoms in the diode are simulated using a tool called molecular dynamics (MD). Statistical analysis of the atoms’ motions is used to compute the thermal conductivity of the diode. Second, an experimental method called photothermal radiometry (PTR) is used to obtain the thermal conductivity through measuring thermal “echoes” from heat put into the diode.

The molecular dynamics results conclude that the thermal conductivity of the QCL across the layers is less than the conductivity of either of the materials that are used to make the laser, which demonstrates that heat transfer is being restricted at the layer
interfaces. In the direction which runs along with the layers, the thermal conductivity is found to be between the values for the two materials that make up the QCL.

The experimental results are not strongly conclusive because the models used in the PTR measurement process do not represent all the heat transfer that is occurring in the samples. Without improved models, the results from the PTR measurements are qualitative at best, but they do demonstrate the presence of the QCL film on the InP wafer. Additional work must be done to better-quantify the results.

While the experimental results are not entirely conclusive, the simulations and experiments corroborate each other sufficiently to warrant further investigation using these techniques. Additionally, the simulations present sufficient internal consistency so as to be useful for thermal property investigation independent of the PTR results.
I dedicate this dissertation and all my doctoral work to my family, but most especially to my wife, Lydia. Through all the difficulties incident to this pursuit she has been my constant companion and friend.
Acknowledgments

- Heng Ban
- Andreas Mandelis
- Anna Matvienko
- Sebastian Volz
- Thomas Antoni
- Mihai Chirtoc
- Robert Spall
- Thomas Hauser
- Bonnie Ogden
- Christine Spall
- Nate Benson
- John Hanks
- Ben Timmins
- Austin Fleming
- Tyson Watkins
- Colby Jensen
- Wassim Kassem
- Haoxue Han
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Public Abstract</td>
<td>v</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>Acronyms</td>
<td>xvi</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>xvii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Quantum Cascade Lasers</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Molecular Dynamics</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Photothermal Radiometry</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Overview of Work</td>
<td>6</td>
</tr>
<tr>
<td><strong>2 Objectives</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>3 Approach</strong></td>
<td>9</td>
</tr>
<tr>
<td>3.1 Molecular Dynamics</td>
<td>9</td>
</tr>
<tr>
<td>3.1.1 LAMMPS</td>
<td>10</td>
</tr>
<tr>
<td>3.1.2 Simulation Systems</td>
<td>11</td>
</tr>
<tr>
<td>3.1.3 Inter-atomic Potentials</td>
<td>14</td>
</tr>
<tr>
<td>3.1.4 Tersoff Potential</td>
<td>16</td>
</tr>
<tr>
<td>3.1.5 Potential Coefficients</td>
<td>20</td>
</tr>
<tr>
<td>3.1.6 Integration Schemes</td>
<td>22</td>
</tr>
<tr>
<td>3.1.6.1 Thermostatting</td>
<td>25</td>
</tr>
<tr>
<td>3.1.6.2 Barostatting</td>
<td>25</td>
</tr>
<tr>
<td>3.1.7 The Brillouin Zone</td>
<td>26</td>
</tr>
<tr>
<td>3.1.8 Dispersion Relations</td>
<td>28</td>
</tr>
<tr>
<td>3.1.9 Green-Kubo Conductivity Calculation</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Photothermal Radiometry</td>
<td>30</td>
</tr>
<tr>
<td>3.2.1 Thermal-wave Field Equation</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Solutions to the TWF Equation</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2.1 1D Solution</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2.2 1D Analytical Solution Stabilization</td>
<td>35</td>
</tr>
<tr>
<td>3.2.2.3 2D Numerical Solution</td>
<td>40</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Averaged cascade layer thicknesses</td>
<td>12</td>
</tr>
<tr>
<td>3.2</td>
<td>Literature coefficients for Tersoff potential</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Linear thermal expansion coefficients from simulation and experiment</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Specific heats from simulation and experiment</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Fit 1</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Fit 2</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison of thermal conductivities with experimental values</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Study of size effects for InAs</td>
<td>77</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary of QCL simulation results</td>
<td>83</td>
</tr>
<tr>
<td>4.8</td>
<td>Uncertainty inputs to Monte Carlo simulation</td>
<td>85</td>
</tr>
<tr>
<td>4.9</td>
<td>Fit results for QCL samples</td>
<td>96</td>
</tr>
</tbody>
</table>
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>General process for molecular dynamics calculations</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Strong and weak scaling performance of LAMMPS on various machines [1]</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Quantum cascade laser diode superlattice on an indium phosphide substrate</td>
<td>11</td>
</tr>
<tr>
<td>3.4</td>
<td>Zinc-blend atomic structure</td>
<td>13</td>
</tr>
<tr>
<td>3.5</td>
<td>QCL single-cascade cross-section</td>
<td>13</td>
</tr>
<tr>
<td>3.6</td>
<td>Probability distribution for AlInAs/GaInAs in the superlattice as a function of position</td>
<td>15</td>
</tr>
<tr>
<td>3.7</td>
<td>Plot of Tersoff bonding energy as a function of intertomic distance for various third-body distances</td>
<td>19</td>
</tr>
<tr>
<td>3.8</td>
<td>Results of simple lattice-dynamics type investigation of the Tersoff potential</td>
<td>19</td>
</tr>
<tr>
<td>3.9</td>
<td>Volume rendering of the bonding environment surrounding an arsenic atom in a indium arsenide zincblend lattice</td>
<td>21</td>
</tr>
<tr>
<td>3.10</td>
<td>Trajectory of a single arsenic atom in indium arsenide from a molecular dynamics simulation using the Tersoff potential</td>
<td>26</td>
</tr>
<tr>
<td>3.11</td>
<td>Brillouin zones for a two-dimensional lattice</td>
<td>27</td>
</tr>
<tr>
<td>3.12</td>
<td>FCC first Brillouin zones for bulk materials and a QCL cascade</td>
<td>29</td>
</tr>
<tr>
<td>3.13</td>
<td>Typical photothermal radiometry (PTR) experimental setup</td>
<td>31</td>
</tr>
<tr>
<td>3.14</td>
<td>Example solutions of the thermal-wave field equation in space and the surface solution verses frequency: a) thermal-wave field in space showing both wave intensity and phase, including a discontinuity from contact resistance; b) surface wave intensity and phase verses frequency</td>
<td>36</td>
</tr>
<tr>
<td>3.15</td>
<td>Example numerical solution of the axisymmetric thermal-wave field equation</td>
<td>40</td>
</tr>
<tr>
<td>3.16</td>
<td>Residual field as a function of thermal diffusivity and effusivity</td>
<td>46</td>
</tr>
<tr>
<td>3.17</td>
<td>Monte Carlo uncertainty evaluation method</td>
<td>47</td>
</tr>
</tbody>
</table>
4.1 Thermal expansion results for AlAs ............................................. 52
4.2 Thermal expansion results for GaAs ............................................. 52
4.3 Thermal expansion results for InAs ............................................. 52
4.4 Thermal expansion results for AlGaAs ......................................... 53
4.5 Thermal expansion results for AlInAs .......................................... 53
4.6 Thermal expansion results for GaInAs ......................................... 53
4.7 Specific heat results for AlAs ..................................................... 56
4.8 Specific heat results for GaAs ..................................................... 56
4.9 Specific heat results for InAs ..................................................... 56
4.10 Specific heat results for AlGaAs .................................................. 57
4.11 Specific heat results for AlInAs .................................................. 57
4.12 Specific heat results for GaInAs .................................................. 57
4.13 Dispersion curves and density of states for InAs as computed by two methods 62
4.14 Dispersion curves and density of states ....................................... 63
4.15 Dispersion curves and density of states with spectral heat flux ............ 72
4.16 Density of states and spectral heat flux ...................................... 73
4.17 Thermal conductivity of AlAs .................................................... 75
4.18 Thermal conductivity of GaAs .................................................... 75
4.19 Thermal conductivity of InAs .................................................... 75
4.20 Thermal conductivity of AlGaAs .................................................. 76
4.21 Thermal conductivity of AlInAs .................................................. 76
4.22 Thermal conductivity of GaInAs .................................................. 76
4.23 Thermal expansion coefficient from QCL simulations ....................... 79
4.24 Specific heat from QCL simulations ............................................. 79
4.25 Dispersion and density of states data for QCL simulations ................ 81
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>Density of States</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public Lisence</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output (Computer activity)</td>
</tr>
<tr>
<td>LAMMPS</td>
<td>Large-scale Atomic/Molecular Massively Parallel Simulator</td>
</tr>
<tr>
<td>MCT</td>
<td>Mercury cadmium telluride (HgCdTe) infrared radiation detector</td>
</tr>
<tr>
<td>MD</td>
<td>Molecular Dynamics</td>
</tr>
</tbody>
</table>
| NTP | Integration that maintains constant:  
  - Number of atoms  
  - Temperature  
  - Pressure |
| NVE | Integration that maintains constant:  
  - Number of atoms  
  - System volume  
  - System energy |
| PN junction | Positively-Negatively doped junction |
| PTR | Photothermal Radiometry |
| QCL | Quantum Cascade Laser |
| SPQ | Spectral Conduction |
Nomenclature

\( \alpha \)  Thermal diffusivity
\( \alpha_L \)  Linear thermal expansion coefficient
\( \alpha_M \)  Morse length scale
\( \mathbf{a}_1 \)  Lattice unit vector
\( \mathbf{a}_2 \)  Lattice unit vector
\( \mathbf{a}_3 \)  Lattice unit vector
\( A \)  Autocorrelation integral mode asymptotic value
\( A \)  Generic quadrupole impedance component
\( A \)  Lock-in real component
\( A \)  Quadrupole impedance term
\( a \)  Generic quadrupole impedance component
\( a \)  Incident beam radius
\( a \)  Lattice parameter
\( A_i \)  Quadrupole impedance term for layer \( i \)
\( A_{ij} \)  Stillinger-Weber 2-body coefficient
\( a_{ij} \)  Stillinger-Weber 2-body coefficient
\( \beta \)  Modified thermal wave number
\( \beta \)  Tersoff bonding environment coefficient
\( \mathbf{b}_1 \)  Lattice unit vector
\( \mathbf{b}_2 \)  Lattice unit vector
\( \mathbf{b}_3 \)  Lattice unit vector
\( B \)  Generic quadrupole impedance component
\( B \)  Lock-in imaginary component
\( b \)  Generic quadrupole impedance component
\( b \)  Ratio of effusivities
\( B_{ij} \)  Stillinger-Weber 2-body coefficient
$b_{ij}$  Tersoff bonding environment multiplier
$C$  Generic quadrupole impedence component
$C$  Volumetric specific heat
$c$  Generic quadrupole impedence component
$c_i$  Tersoff 3-body coefficient
$C_P$  Specific heat at constant pressure
$C_V$  Specific heat at constant volume
$\Delta t$  Time step size
$\delta$  Hankel-space radial position
$\nabla$  Del vector operator
$D$  Generic quadrupole impedence component
$D$  Tersoff cutoff length scale
$d$  Generic quadrupole impedence component
$d_i$  Tersoff 3-body coefficient
$D_M$  Morse dimmer energy
DOS  Density of states
$\epsilon$  Lennard-Jones energy scale
$\epsilon$  Barostat dynamics modification term
$\epsilon$  Thermal effusivity
$\epsilon_{ijk}$  Stillinger-Weber 3-body coefficient
$\epsilon_{ij}$  Stillinger-Weber 2-body coefficient
$E$  System total energy
$E'$  Stablized quadrupole impedance term
$E_i$  Hankel-space quadrupole impedence matrix term for layer $i$
$F$  Fourier transform
$F_{0i}$  Fourier function for layer $i$
$\Phi$  Hankel-space heat flux phasor
$\phi$  Lock-in phase
\[ \phi \] Phasor heat flux

\[ \Phi_0 \] Hankel-space heat flux phasor at \( z=0 \)

\[ \phi_0 \] Phasor heat flux at \( x=0 \)

\[ \phi_2 \] Stillinger-Weber 2-body potential term

\[ \phi_3 \] Stillinger-Weber 3-body potential term

\[ \Phi_b \] Hankel-space heat flux phasor at sample back

\[ \phi_b \] Phasor heat flux at layer back of sample

\[ \Phi_f \] Hankel-space heat flux phasor at sample front

\[ \phi_f \] Phasor heat flux at layer front of sample

\[ \Phi_i \] Hankel-space heat flux phasor at layer \( i \)

\[ \phi_i \] Phasor heat flux at layer \( i \)

\[ F \] Quadrupole impedance term

\[ f_A \] Fitting function for autocorrelation

\[ f_A \] Tersoff attractive function

\[ f_C \] Tersoff cutoff function

\[ F_i \] Quadrupole impedance term for layer \( i \)

\[ f_I \] Fitting function for autocorrelation integral

\[ f_R \] Tersoff repulsive function

\[ K' \] Stablized quadrupole impedance term

\[ \gamma_{ij} \] Stillinger-Weber 3-body coefficient

\[ g \] Tersoff bonding environment function

\[ \eta \] Thermostat dynamics modification term

\[ \mathcal{H} \] Atomic Hamiltonian

\[ \mathcal{H} \] Hankel transform

\[ I \] Dispersion intensity

\[ i \] Imaginary unit \( i = \sqrt{-1} \)

\[ I_0 \] Incident heating power

\[ \mathbf{J} \] System heat flux
$k$  Wave-vector

$K$  Quadrupole impedance term

$k$  Thermal conductivity

$k_B$  Boltzmann constant

$K_i$  Quadrupole impedance term for layer $i$

$\lambda_1$  Tersoff repulsion length scale

$\lambda_2$  Tersoff attraction length scale

$\lambda_3$  Tersoff 3-body length scale

$\lambda_{ijk}$  Stillinger-Weber 3-body coefficient

$\mathcal{L}$  Laplace transform

$L$  Layer thickness

$L$  Lock-in amplifier signal

$L_i$  Layer thickness for layer $i$

$\mu$  Thermal diffusion length

$M$  System total mass

$m$  Atomic mass

$m$  Tersoff bonding environment coefficient

$M$  Number of time steps in simulation

$n$  Normal vector

$N$  Number of atoms in simulation

$n$  Tersoff bonding environment coefficient

$\omega$  Radial frequency (rad/s)

$p$  Atomic momentum

$p$  Autocorrelation approach order

$p$  Quadrupole off-diagonal term

$p_i$  Quadrupole off-diagonal term for layer $i$

$p_{ij}$  Stillinger-Weber 2-body coefficient

$q$  Atomic position
Atomic heat flux for atom $i$

$q_i$  Thermal wave number

$q_i$  Thermal wave number for layer $i$

$q_{ij}$  Stillinger-Weber 2-body coefficient

$R$  Lock-in magnitude

$R$  Number of simulations in the ensemble

$R$  Tersoff mean cutoff distance

$r$  Inter-atomic distance

$r$  Radial position

$r_0$  Morse spacing offset

$R_i$  Inter-layer thermal resistance $R$ at position $i$

$r_m$  Modified Lennard-Jones length scale

$r_{ij}$  Inter-atomic distance between atoms $i$ and $j$

$r_{ik}$  Inter-atomic distance between atoms $i$ and $k$

SPQ  Spectral conduction

$\sigma$  Lennard-Jones length scale

$\sigma_{ij}$  Stillinger-Weber 2-body coefficient

$S$  Input signal to lock-in amplifier

$S$  Normalizing scale factor

$S$  Structure factor

$S_p$  Sensitivity to parameter $p$

$\tau$  Autocorrelation characteristic time scale

$\tau_p$  Barostat characteristic time

$\tau_T$  Thermostat characteristic time

$\Theta$  Hankel-space temperature phasor

$\Theta_0$  Hankel-space temperature phasor at $z=0$

$\Theta_f$  Hankel-space temperature phasor at sample front

$\Theta_i$  Hankel-space temperature phasor at layer $i$
\( T \) Temperature
\( t \) Time
\( t_j \) Discrete time \( j \)
\( U_i \) Hankel-space quadrupole impedance matrix term for layer \( i \)
\( \angle \theta \) Phase of temperature \( \theta \)
\( \cos (\theta_{0i}) \) Tersoff 3-body coefficient
\( \mathbf{v} \) Atomic velocity
\( \mathbf{v}_g \) Phonon group velocity
\( \Theta \) Laplace-space phasor temperature
\( \theta \) Phasor temperature
\( \theta_0 \) Phasor temperature at \( x=0 \)
\( \Theta_b \) Hankel-space temperature phasor at sample back
\( \theta_b \) Phasor temperature at layer back of sample
\( \theta_f \) Phasor temperature at front of sample
\( \theta_i \) Phasor temperature at layer \( i \)
\( \theta_{0ijk} \) Stillinger-Weber 3-body coefficient
\( \theta_{ijk} \) Inter-atomic angle made by atoms \( i, j \) and \( k \)
\( V \) System volume
\( V_e \) Inter-atomic potential energy
\( x \) Atomic position
\( \xi \) Laplace-space position
\( x \) Generalized degree of freedom
\( x \) Position
\( [Z] \) Quadrupole impedance matrix
\( [Z'] \) Stabilized quadrupole impedance matrix
\( \zeta \) Laplace-space depth position
\( \zeta_{ij} \) Tersoff bonding environment coefficient
\( z \) Sample depth position
Chapter 1

Introduction

1.1 Quantum Cascade Lasers

First constructed in 1994 by Faist et al. [2], quantum cascade laser (QCL) devices presented a completely new form of laser technology originally proposed more than two decades prior by Kazarinov and Suris [3]. Conventional diode lasers generate photons through the radiative recombination of a conduction electron and a valence-band hole injected through a forward-biased PN junction [4, 5]. QCLs do not rely on the recombination of an electron and hole, but rather form a series of drops in band energies that the electrons must pass through under bias. At each drop, the electron must emit energy in the form of a photon to enter the next band, which results in a single electron releasing a number of photons during its path through the device.

The energy drops are created using intersubband transitions in a superlattice-like structure that create a stack of quantum well heterostructures. The structure is not a superlattice in the traditional sense, since the layer thicknesses are not fully periodic, but are chirped, having a progression from thicker to thinner in a short series that is repeated. The alternating layers of the superlattice have been constructed from alloys containing aluminum, gallium, and indium arsenides, as well as some antimonide compounds. These materials are used since the band energy engineering can be accomplished most effectively with them. A single repetition of the chirped pattern is called a cascade. There are as many as seventy cascades in the active region of a QCL, and the energy drops in the cascades have been compared to the various levels of water in a cascading waterfall. Despite the significant electron and hole mobility within the structure under bias, the dominant heat carrier of the materials are still phonon vibrations within the lattice.

After their inception, most work with QCLs focused on increasing their power output
[6, 7], manufacture techniques and verification [8] and efficiency [9]. A major aspect of all laser operation, including QCLs, is thermal management of the device. Current passing through the diode creates heat, which raises the temperature of the diode and decreases the efficiency. Efforts to confront these challenges first focused on characterizing the thermal performance of QCL diodes in terms of thermal properties and temperature dependence in output and threshold current. More recently efforts have transitioned into consideration of appropriate materials into which the device may be embedded to provide superior heat removal [10–12].

Spagnolo et al. [13] were among the first to measure the thermal conductivity of a QCL’s active region, which they accomplished by temperature measurement through the thermal shift of the band-to-band photoluminescence peaks in the AlInAs and InP cladding layers of the QCL; their results predict the thermal conductivity across a range of temperatures, but never exceeding 8 W/m·K. Lops et al. [14] reported 2 W/m·K, and Vitiello et al. [15–17] reported similar results from photoluminescence-based measurements. Such experiments were also conducted by Scamarcio et al. [18], who report values from 1.8 W/m·K to 6 W/m·K. Groups headed by Evans [19] and Spagnolo [20] both developed conductivity estimates by correlating finite element analysis (FEA) of the heat transfer in QCLs with experimental measurements. Recently, Vitiello’s group has also measured the thermal diffusivity of QCLs [21].

QCLs have many potential applications, including non-invasive sub-millimeter detection systems for security [22, 23]. Other applications capitalize on the tunability of the diodes; this allows the exact output wavelength to be adjusted during operation [24, 25], which can be used for chemical detection systems.

1.2 Molecular Dynamics

The method of molecular dynamics (MD) simulation comes from a simple idea: that if the forces on all the atoms in a material were known, then they could be integrated in time, and all the properties computed from these atomic trajectories. Since there are numerous integration algorithms with various attractive properties, the greatest difficulty
in this process is the modeling of the forces between atoms. The forces acting between atoms are normally expressed as the local gradient of the inter-atomic energy, called the potential.

While the inter-atomic potential can be computed from quantum mechanics, frequently termed \textit{ab initio} molecular dynamics, such calculations are computationally expensive. The cost of \textit{ab initio} simulation is so high that many practical problems cannot be handled with current computing capabilities. This dilemma has resulted in the development of empirical potentials which can capture many significant material properties and behaviors without the need for expensive quantum mechanical calculations. There are two disadvantages to these potentials: 1) they cannot exactly match experimental results for material properties (which is not guaranteed for \textit{ab initio} simulations either) and 2) that they rely on the fitting of coefficients to experimental data.

Empirical potentials have been used almost from the onset of MD’s application to problems of interest, and a great diversity of potential forms have been devised for various bonding types and material configurations. The most basic model in common use is the Lennard-Jones [26] (LJ) potential, which is used to model van der Waals bonds. This potential has application to a variety of physical systems, including many low-temperature crystals such as xenon and argon [27, 28], as well as the forces between sheets of graphene. Because of its computational simplicity and general understanding [29], the LJ potential is often used to demonstrate new MD software [30] and as a normalization for the computational cost of other potentials.

The development of empirical potentials can be separated into functional development and coefficient fitting. The functional for a potential is the mathematical form that the potential takes, and usually contains parameters or coefficients that can be used to adjust the potentials behavior. The functional’s form is generally selected to provide flexible support for various physical phenomenon, such as two- or three-body interactions, long-range effects, ionic charges, etc. The coefficients are selected by fitting the properties predicted by MD simulation with the functional to experimentally determined values. Thus,
a single functional can be used to simulate various materials through different coefficient sets.

The simulation of semiconductor materials such as those contained in QCLs requires more sophisticated physics than the van der Waals forces, and has led to the development of numerous functional forms and potential coefficient sets. While there exist variations among the specific notation used to describe the functional forms used, most semiconductor simulations implement either the Stillinger-Weber or Tersoff potentials, or a minor modification of one of these.

The Stillinger-Weber [31] (SW) model was first developed to simulate silicon, and has seen much use [28,32–34] for silicon simulations because of its simplicity and accuracy [35]. Initially the SW functional was extended to germanium with new coefficients [36], and then to interactions including both silicon and germanium in the same system [37,38].

The Tersoff model was also developed initially for silicon [39–41], and then extended to germanium and carbon, as well as silicon carbide and silicon germanium [42]; it has even been applied to the simulation of hydrocarbons [43] and zinc selenide [44]. When compared with other potentials for the simulation of semiconductors it was found to perform as well as SW for many materials and properties [35, 45]. The variety of materials which can be successfully simulated using the Tersoff potential is partly due to the three-body nature of the functional. Many potentials only consider interacting atoms, but Tersoff added terms which include the effects of atoms nearby this interaction, called the bonding environment. Thus, angular effects and three-body interactions can be included in coefficient sets for the Tersoff functional.

An interesting aspect of Tersoff potential for the simulation of compounds is the possibility to create new coefficients for mixed interactions from the coefficients for the respective elemental interactions. Partially due to this feature, the Tersoff potential has become the most commonly used functional form for the simulation of III-V compounds used in laser diode manufacture. Since these efforts have resulted in the development of good coefficient sets to simulate many III-V materials, the Tersoff potential was selected for the QCL sim-
ulations in the present work. This justifies a more complete examination of the potential
coefficient developments in the literature.

The materials of interest for QCL simulation are compounds composed of aluminum,
gallium, indium and arsenic. The first of these materials to be simulated using the Tersoff
potential were gallium and arsenic, for the simulation of gallium arsenide. Smith [46]
developed coefficients for gallium arsenide after concluding that the earlier work of Ito [47]
was inadequate for atomic collision calculations. These coefficients became the foundation of
all Tersoff coefficients used to simulate gallium arsenide for the better part of a decade. Ashu
et al. [48] used Smith’s coefficients for Ga-Ga and As-As interactions, but created new fits
for the Ga-As interactions after concluding that Smith’s values for Ga-As were inadequate.
In the same work, Ashu also created new fits for In-In, and In-As, by fitting to dimer energies
and bond lengths, cohesive energies, cluster formation energies and elastic constants. The
interactions for In-Ga were constructed using the mixing model first proposed by Tersoff [42].

Largely independent of Ashu’s work, Sayed et al. [49,50] also took Smith’s coefficients
as a basis, but re-fit the coefficients for Ga-As and added new coefficients for Al-Al and
Al-As in their simulation of atomic displacements. A comprehensive consideration of the
coefficients fitted by Smith, Ashu and Sayed was performed by Nordlund et al. [51] for the
simulation of mixing at semiconductor interfaces. New coefficients were fit for the Ga-As,
Al-As and In-As interactions due to the various insufficiencies of the previous sets. The
present work mostly uses the coefficients from Nordlund, but some with modifications as
discussed in a later section.

1.3 Photothermal Radiometry

Initially developed by Nordal and Kanstad [52,53] as a spectroscopy method, photother-
mal radiometry (PTR) has become a standard means of measuring thermal properties for
layered samples using a variety of configurations and methods. The method consists of
thermally exciting the surface of a sample with a laser and measuring the change in emitted
radiation which results from the heating. While Nordal and Kanstad stayed with the
spectroscopy aspects of PTR, other researchers rapidly extended the measurement method
to scanning samples in excitation frequency instead of laser wavelength. Both methods can be used to analyze the optical and thermal properties of a sample, but most measurements done today use a variety of the frequency scan method.

The connection between photothermal radiometry and other photothermal methods is very strong, and several photothermal methods are often used to characterize the same sample so as to take advantage of each methods’ strengths [54,55]; there are even variations of method within the bounds of PTR including a pulsed variety [56–58], instead of standard modulation, as well as more exotic methods [59–68]. Many researchers analyzed the heat transfer conditions in the various experiments, resulting in a wealth of analytical expressions for the various conditions which arise in photothermal measurements [69–72]. Photothermal methods have also been applied with atomic force microscopy (AFM) and scanning thermal microscopy (SThM) to allow for simultaneous measurement of the topological and thermal aspects of a sample [73, 74]. Another application, that saw almost immediate use, is the detection of subsurface defects and evaluation of material quality through the use of various photothermal methods, including PTR [56,75–79].

Perhaps a good measure of the influence held by photothermal methods in general, and photothermal radiometry in particular, is the existence of multiple introduction and overview papers [80–82] written about them and intended to familiarize other researchers with the advantages incident to their application. The method can even be applied to organic systems for food evaluation [83] and medical applications [84, 85]. There also exist many papers detailing the measurement of various samples using PTR [86–93].

1.4 Overview of Work

In the proposed work, molecular dynamics is used to simulate a single chirped period of a quantum cascade laser. To this end, literature values for the needed coefficients in the Tersoff potential have been found; before serious use, however, these coefficients were validated against experimental data. The predicted thermal conductivity of each III-V pairing and alloy in the final QCL has been checked against experimental values. The phonon dispersion curves have been computed and compared with what experimental data
is available.

A photothermal radiometry system has been constructed at Ecole Central de Paris and its operation tested. QCL samples have been measured and their frequency scan results analyzed to compare with the simulation results. The uncertainty of PTR systems in general has been considered using the Monte Carlo uncertainty method.
Chapter 2

Objectives

1. Use molecular dynamics to compute the effective thermal conductivity of a characteristic quantum cascade laser diode super-lattice structure.
   - Evaluate inter-atomic potential by simulation and conductivity calculation of III-V bulk materials in QCL composition and alloys thereof.
   - Compute the cross-plane QCL thermal conductivity.
   - Compute the in-plane QCL thermal conductivity.
   - Compare results with literature.

2. Construct a photothermal radiometry system.
   - Verify correct operation of system.
   - Assess the uncertainty trends in PTR systems using Monte Carlo uncertainty analysis.
   - Run frequency scans on QCL; interpret results.
Chapter 3
Approach

3.1 Molecular Dynamics

In this work, molecular dynamics simulations were used to compute the expected thermal properties of nanoscale structures, specifically the chirped superlattice period present in quantum cascade laser devices. Because MD simulations attempt to duplicate the motion of all the atoms in a system, the desired properties must be extracted from the results of the simulation and are not directly computed as they would be in a more traditional numerical model such as finite element analysis. The general process follows that show in Figure 3.1.

The system is setup and initialized, which involves several steps. All atoms in the simulation must be created, and initial velocities assigned. Atoms are placed at lattice sites according to the material to be simulated, and velocity initialization is accomplished by a randomized distribution for a particular temperature. This process requires a seed value for the random number generator to ensure that each simulation represents a unique sampling of phase space. Additionally, the interatomic potential must be assigned at this stage.

Immediately after initialization the system is not in a natural state, and two important steps must be taken before the time evolution of the system is physically significant. First, the system must be relaxed to the potential. This can be done by minimizing the
system’s potential energy, or alternatively, by simulating the system while enforcing a constant pressure and temperature but allowing the volume and energy to vary. The latter approach is used in the present work because of its simplicity. The second important step is the equilibration of the system, which allows the statistical distributions within the system to reach their equilibrium forms. Since all later calculations assume the system fluctuates about equilibrium, their results are invalid without this step.

Once the system has been setup, relaxed, and equilibrated, it can be run for a time during which the atomic trajectories can be considered to represent part of the equilibrium ensemble. At this stage various system properties can be saved to files on disk for later processing, or statistics can be computed as the simulation runs. Either way, there is often significant work required to extract the desired parameters from the atomic trajectories.

3.1.1 LAMMPS

While there exist many molecular dynamics programs which could be used for the simulation of quantum cascade lasers, the LAMMPS software from Sandia National Laboratories was selected. The decision to use LAMMPS was motivated by several factors, including its flexible operation, open license, and excellent scalability. Designed from the start to tackle the largest problems on the biggest machines, LAMMPS requires the message passing interface (MPI) library for operation, even when executing in serial mode. Figure 3.2 [1] shows the scaling performance of LAMMPS on various national lab cluster computers for both strong and weak scaling. Strong scaling keeps the system size fixed (constant atom count) and simulates on increasing numbers of nodes/cores. Weak scaling increases the atom count so that the atoms per core are held constant. The strong scaling of a parallel algorithm demonstrates how much faster a particular simulation can be made by adding more cores in parallel, while the weak scaling performance demonstrates the size of simulations which can be efficiently run with the given resources.

LAMMPS is licensed under the general public license (GPL), which ensures that the program’s source code is available for modification if needed, but also that any derivative works be released under the same license. This stipulation ensures that the code remains
open source, and encourages collaboration between the various groups using the software. LAMMPS also encourages flexibility in its operation by the use of input scripts which allow an operator nearly full control over a simulation; this makes it possible to simulate many different systems and compute a variety of properties using the same code.

### 3.1.2 Simulation Systems

Like many solid-state devices, quantum cascade lasers are constructed by depositing the appropriate structures atop a lattice-matched wafer. QCLs are almost always created using molecular-beam epitaxy (MBE), which allows for very fine control over layer thickness as the chirped superlattice is laid down. A diagram of a QCL on its substrate can be seen in Figure 3.3.

![Fig. 3.3: Quantum cascade laser diode superlattice on an indium phosphide substrate](image)

Fig. 3.2: Strong and weak scaling performance of LAMMPS on various machines [1]
At the atomic level, the III-V semiconductor materials, from which most QCLs are composed, form zinc-blend lattices as shown in Figure 3.4. This geometry dictates the lattice on which the atoms are placed during system construction for the molecular dynamics simulations, which can be expressed as two overlapping face-centered-cubic (FCC) lattices. While it is possible to construct the system as a single FCC lattice with a two-atom basis, the algorithms work out more easily to build it as two overlapping FCC lattices.

Table 3.1 lists an averaged cascade, based on the cascades of an entire infrared wavelength QCL. The layers alternate between GaInAs and AlInAs in order to create the needed intersubband energy transitions. Some of the layer thicknesses are on the same order as the lattice constant for these materials (approximately 6 Å), and none are exact multiples of the lattice constant. This is possible since the layers listed are the design intents, not actual thicknesses. The layer specification corresponds to the epitaxial procedure, while the actual lattice is less definite, as partial layers laid down mix with subsequent layers in the structure. The layer thicknesses are correct in an averaged sense across the entire layer, but locally they vary slightly as the adjacent layer mixed during formation.

When the full single-cascade period is constructed, a small cross-section of the simulation looks like Figure 3.5. The layers are clearly visible in the x-direction, although the periodic boundaries cause the atoms to apparently 'jump' from one side of the simulation domain to the other. While this makes it difficult to visualize the simulation results, it is key to mimicking bulk behavior.

The system is constructed by populating the lattice a single atom at a time. One

<table>
<thead>
<tr>
<th>#</th>
<th>Composition</th>
<th>Thickness Å</th>
<th>#</th>
<th>Composition</th>
<th>Thickness Å</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GaInAs</td>
<td>35.8</td>
<td>9</td>
<td>GaInAs</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>AlInAs</td>
<td>14.1</td>
<td>10</td>
<td>AlInAs</td>
<td>14.6</td>
</tr>
<tr>
<td>3</td>
<td>GaInAs</td>
<td>34.4</td>
<td>11</td>
<td>GaInAs</td>
<td>47.7</td>
</tr>
<tr>
<td>4</td>
<td>AlInAs</td>
<td>18.5</td>
<td>12</td>
<td>AlInAs</td>
<td>8.4</td>
</tr>
<tr>
<td>5</td>
<td>GaInAs</td>
<td>32.6</td>
<td>13</td>
<td>GaInAs</td>
<td>51.5</td>
</tr>
<tr>
<td>6</td>
<td>AlInAs</td>
<td>25.7</td>
<td>14</td>
<td>AlInAs</td>
<td>9.7</td>
</tr>
<tr>
<td>7</td>
<td>GaInAs</td>
<td>30.8</td>
<td>15</td>
<td>GaInAs</td>
<td>44.2</td>
</tr>
<tr>
<td>8</td>
<td>AlInAs</td>
<td>37.6</td>
<td>16</td>
<td>AlInAs</td>
<td>12</td>
</tr>
</tbody>
</table>
Fig. 3.4: Zinc-blend atomic structure

Fig. 3.5: QCL single-cascade cross-section
of the two FCC lattices is only populated by arsenic, since all the III-V compounds used in the QCL are arsenic-based and it was deemed very unlikely for arsenic to occupy a position for a group III element or the reverse. The second FCC lattice sites’ occupancy is determined by a probability distribution function shown in Figure 3.6. The figure also displays a representative slice of the QCL system showing the composition of each layer in the chirped period.

As can be seen in Figure 3.6, the probability distribution function for the lattice site occupations has been rounded to what would be sharp definitions in table 3.1. This was accomplished by first defining the probability distribution as step functions at layer boundaries, and then convolving the distribution with a Gaussian function with a characteristic width of 3 Å (about half a unit cell). By smoothing the occupation distribution, the effects of short-range diffusion at the layer boundaries were able to be approximated.

3.1.3 Inter-atomic Potentials

There exist a variety of empirical inter-atomic potentials which can be used for the simulation of molecular systems, and only a few are considered here due to their increased relevance to the present work. The most fundamental potential is that created by Lennard-Jones (LJ) for the simulation of systems dominated by van der Waals forces. The standard form is found in Equation 3.1, while using a revised length parameter in 3.2, the form can be modified to 3.3.

\[
V_e = 4\epsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right] \quad (3.1)
\]

\[
r_m = 2^{1/6} \sigma \quad (3.2)
\]

\[
= \epsilon \left[ \left( \frac{r_m}{r} \right)^{12} - \left( \frac{r_m}{r} \right)^6 \right] \quad (3.3)
\]

Sometimes called the LJ12-6 potential because of the powers on the repulsive and attractive terms, the potential still sees heavy use by the molecular dynamics community. The potential has proven effective at simulating the liquid and solid phases of most of the noble gases, including xenon and argon. There are two fitting parameters or coefficients used
Fig. 3.6: Probability distribution for AlInAs/GaInAs in the superlattice as a function of position

in the potential, $\epsilon$ and $\sigma$, which are energy and length scales respectively. The coefficients can be computed from measurements of gas phase viscosity and virial coefficients, and are accurately known for many elements [94].

Because of its ubiquity, the Lennard-Jones potential is often used as a benchmarking tool to compare the performance of molecular dynamics software, but also to compare the computational expense of various potentials in a particular program. For example, LAMMPS provides comparative performance metrics for all implemented potentials, but does so by normalizing the performance to the Lennard-Jones potential wall-times.

The Morse potential was first developed from solutions to the Schroedinger equation for diatomic molecules with an assumed potential energy function, and was proven successful in predicting vibrational energy levels and electron level data. This rigorous background has made the Morse potential an attractive starting point for the development of other, more empirical, potentials that by deviation from analytical solutions can model more complex behaviors. While originally derived in the form of Equation 3.4, LAMMPS implements the potential as Equation 3.5 without the constant offset.

$$ V_e = D_M [1 - \exp (-\alpha_M (r - r_0))]^2 $$

$$ = D_M [\exp (-2\alpha_M (r - r_0)) - 2 \exp (-\alpha_M (r - r_0))] + D_M $$
The Stillinger-Weber (SW) potential was first conceived by expressing interatomic potentials in terms of one, two and three body interactions. The one body interactions can be used to express boundary conditions and body forces acting on the system; these terms are implied in all potentials. While the functional forms were largely chosen for their flexibility and natural cutoff, they have proven successful in the simulation of silicon and some other materials. The full potential is given in Equation 3.6, but is composed of the two and three body terms defined in equations 3.7 and 3.8, respectively. The two body terms are similar in form to the morse potential, while the three body terms include angular dependence as well.

\[
V_e = \sum_i \sum_{j>i} \phi_2 (r_{ij}) + \sum_i \sum_{j \neq i} \sum_{k>j} \phi_3 (r_{ij}, r_{ik}, \theta_{ijk}) \tag{3.6}
\]

\[
\phi_2 (r_{ij}) = A_{ij} \epsilon_{ij} \left[ B_{ij} \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{p_{ij}} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{q_{ij}} \right] \exp \left( \frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}} \right) \tag{3.7}
\]

\[
\phi_3 (r_{ij}, r_{ik}, \theta_{ijk}) = \lambda_{ijk} \epsilon_{ijk} \left[ \cos \theta_{ijk} - \cos \theta_{\theta_{ijk}} \right]^2 \times \exp \left( \frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}} \right) \exp \left( \frac{\gamma_{ik}\sigma_{ik}}{r_{ik} - a_{ik}\sigma_{ik}} \right) \tag{3.8}
\]

Some researchers consider a major advantage of the Stillinger-Weber potential over that of Tersoff to be its better performance, but as implemented in LAMMPS, the Tersoff potential outperforms the Stillinger-Weber by approximately 13% and enjoys better parallel speedup as well [1].

### 3.1.4 Tersoff Potential

The Tersoff potential, initially proposed by its namesake in 1985, represents bonding energy as the difference of an attractive and repulsive potential as shown in Equation 3.9; many potentials take this form, but Tersoff specifically based this choice on the success of the Morse potential’s form in computation of the binding energy curves for solid cohesion. The cutoff function, as well as the attractive and repulsive terms are given by equations 3.10 through 3.12. Equations 3.13 through 3.15 detail the implementation of the three body
component of the potential.

\[ V_e = \frac{1}{2} \sum_i \sum_{i \neq j} \left[ f_C (r_{ij}) (f_R (r_{ij}) - b_{ij} f_A (r_{ij})) \right] \] (3.9)

\[
f_C (r) = \begin{cases} 
1 & r < R - D \\
\frac{1}{2} - \frac{1}{2} \sin \left( \frac{\pi}{2} \frac{r - R}{D} \right) & R - D < r < R + D \\
0 & r > R + D
\end{cases}
\] (3.10)

\[ f_R (r) = A \exp (- \lambda_1 r) \] (3.11)

\[ f_A (r) = B \exp (- \lambda_2 r) \] (3.12)

\[ b_{ij} = (1 + \beta^n \zeta_{ij})^{\frac{1}{m}} \] (3.13)

\[ \zeta_{ij} = \sum_{k \neq i,j} \left[ f_C (r_{ik}) g (\theta_{ijk}) \exp (\lambda_3^n (r_{ij} - r_{ik})^{m}) \right] \] (3.14)

\[ g (\theta_{ijk}) = 1 + \frac{c_i^2}{d_i^2} - \frac{c_i^2}{d_i^2 + (\cos \theta_{ijk} - \cos \theta_{0i})^2} \] (3.15)

The main expression includes exponential decay functions for the attractive and repulsive forces, as well as a cutoff function which serves to force the potential to be effective at short ranges only, thereby allowing simulations to ignore distant atoms when computing interatomic forces. The attractive potential is multiplied by the term \( b_{ij} \), which implicitly contains the bond order and thereby must depend on the local bonding environment. This term is thus defined as a function of nearby atoms besides the two involved in the currently-considered bond. For this reason the Tersoff potential is often called a three-body potential since it includes terms which consider more than two atoms. However, unlike the Stillinger-Weber potential, the Tersoff potential does so by modifying the bond order implicitly expressed in \( b_{ij} \) to account for the presence of nearby atoms instead of including an additional explicit three-body potential term. One special note must be made about the \( \cos \theta_{0i} \) coefficient; while the form would seem to assert that the value must be bounded such that \( \cos \theta_{0i} \in [-1, 1] \), experimentally fitted coefficient sets frequently result in values outside this range, causing some researchers to denote the coefficient as \( h_i \) [95].

Figure 3.7 shows the Tersoff bond energy for a pair of atoms under the influence of a
third atom at various distances. It can be seen from the plot that there exists no energy minimum if the third atom is too close, but as it moves further away a definite minimum forms. Also visible in the plot are the effects of the cutoff function set to a range of about 3.5 Å, where all the lines abruptly shift to zero instead of asymptotically approaching it. This cutoff distance is generally used to include first neighbors but avoid second neighbors. Also, as can be seen from the potential definition, the cutoff is defined by a sine function to ensure as smooth a transition as possible.

To better understand the properties of the Tersoff potential, a simple lattice dynamics code was written which implements the Tersoff potential for a small neighborhood of atoms in a zinc blend structure, such as those depicted in Figure 3.4. The quantitative results of the investigation are unimportant, but the investigation was able to demonstrate two important results. The first, shown in Figure 3.8a, verifies that the potential does indeed predict a particular lattice parameter for sets of coefficients similar to those used for III-V semiconductors. The second result is shown in Figure 3.8b, which plots the energy and energy gradient along a particular path though 3D space. The important conclusion here is that while the force function is linear near the equilibrium position, it deviates from this form as the atom is displaced progressively further from equilibrium. A linear relationship between interatomic force and position is called harmonic bonding, and is often assumed in material analysis. By extension, a nonlinear relationship is called anharmonic, and causes any results based on the harmonic assumption to be inaccurate. Since the coefficients used for the semiconductors in QCLs predict anharmonic interactions, more traditional lattice dynamics simulations such as GULP cannot be used to adequately analyze these systems and one is forced to use full molecular dynamics simulations.

The simple lattice dynamics exploration of the Tersoff potential was taken to its logical end and a fully three dimensional mapping of the bond energy was calculated for a single arsenic atom in indium arsenide. A volume rendering of the potential is included in Figure 3.9, has been colorized such that regions of high energy are shown as transparent red, while regions of low energy are opaque blue. In this way, the bonding environment can be
Fig. 3.7: Plot of Tersoff bonding energy as a function of intertomic distance for various third-body distances

(a) Energy verses lattice parameter
(b) Bond anharmonicity in Tersoff potential

Fig. 3.8: Results of simple lattice-dynamics type investigation of the Tersoff potential
3.1.5 Potential Coefficients

All empirical potentials like that of Tersoff require coefficients for the functional form in order to model interatomic potentials, which is perhaps the most difficult aspect of their use. Most research utilizing molecular dynamics simulations relies on coefficients from the literature, thereby avoiding the need to create new coefficients. The literature coefficients for interactions between all the materials in the QCLs considered are listed in table 3.2.

It is immediately apparent from the table that many coefficients are required to simulate such a complex system as a full quantum cascade period. All empirical potential coefficients are developed by fitting molecular dynamics simulations’ predicted material properties to experimentally determined values. Often these experimental values are stiffness coefficients, lattice parameters, bonding energies and others. The fitting process is non-trivial, and often requires special care to guide the fitting algorithms. The entire process is burdened by the high computing cost of molecular dynamics simulations, since the objective function of the fit (which would include multiple MD simulations) must be evaluated numerous times to reach a good result.

The coefficients in table 3.2 were largely taken from the work of Nordlund [51], who took many coefficients from Smith [46], Sayed [49,50] and Ashu [48]. Others were modified by Nordlund to correct perceived problems with the results of their use. Recently, Powell has systematically evaluated the III-V materials simulated by the Tersoff potential with a variety of coefficient sets [96]. There are presently no coefficients in the literature for interactions between aluminum and indium; the values appearing in table 3.2 are the results of mixing the elemental coefficients of the respective materials using mixing rules first proposed by Tersoff [42]. Problems are anticipated with these interactions; this practice seems to be the standard starting point for coefficients to predict the interatomic forces for mixed atomic arrangements. When comparing literature values for mixed coefficients with those predicted by the Tersoff mixing rule, most coefficients were within 10% of the mixing rule value, and the maximum deviation was only 60%.
Fig. 3.9: Volume rendering of the bonding environment surrounding an arsenic atom in a indium arsenide zincblend lattice

Table 3.2: Literature coefficients for Tersoff potential

<table>
<thead>
<tr>
<th>Term</th>
<th>Units</th>
<th>III-III</th>
<th>III-V</th>
<th>Elemental</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Al-Ga</td>
<td>Al-In</td>
<td>In-Al</td>
</tr>
<tr>
<td>$R$</td>
<td>Å</td>
<td>3.50</td>
<td>3.55</td>
<td>3.55</td>
</tr>
<tr>
<td>$D$</td>
<td>Å</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>$A$</td>
<td>eV</td>
<td>699.82</td>
<td>1210.89</td>
<td>1210.89</td>
</tr>
<tr>
<td>$B$</td>
<td>eV</td>
<td>55.99</td>
<td>91.13</td>
<td>91.13</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>1/Å</td>
<td>2.55</td>
<td>2.60</td>
<td>2.60</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>1/Å</td>
<td>1.21</td>
<td>1.30</td>
<td>1.30</td>
</tr>
<tr>
<td>$\alpha$</td>
<td></td>
<td>4.60</td>
<td>6.09</td>
<td>3.40</td>
</tr>
<tr>
<td>$\beta$</td>
<td></td>
<td>0.27</td>
<td>0.32</td>
<td>2.11</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>1/Å</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>$m$</td>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td>0.08</td>
<td>0.07</td>
<td>0.08</td>
</tr>
<tr>
<td>$d$</td>
<td></td>
<td>19.27</td>
<td>19.57</td>
<td>19.26</td>
</tr>
<tr>
<td>$\cos \theta_0$</td>
<td></td>
<td>3.24</td>
<td>-0.66</td>
<td>7.39</td>
</tr>
</tbody>
</table>
After careful evaluation of the results from the mixed coefficients created for the interactions between In-Al and Al-In, it was decided to leave them as predicted by Tersoff’s mixing model. The rational for this decision is that the raw mixed coefficients result in the most stable systems, and are modifications along the lines of those seen in the literature only decreased stability without improving the accuracy of the thermal conductivity predictions.

3.1.6 Integration Schemes

Given its foundation in physics and computational chemistry, it should come as no surprise that the methods of molecular dynamics are formulated using the Hamiltonian formalism to derive the equations of motion. The dynamics of a system can be expressed in terms of the Hamiltonian in the form of equations 3.16 and 3.21.

\[
\frac{dp}{dt} = -\frac{\partial H}{\partial q} \quad (3.16)
\]

\[
\frac{dq}{dt} = \frac{\partial H}{\partial p} \quad (3.17)
\]

The Hamiltonian of a system is the function which expresses the total energy of that system in phase space, which for classical mechanics problems such as molecular dynamics, can be taken as the Cartesian position and momentum vectors as show in 3.18 and 3.19.

\[
q = x \quad (3.18)
\]

\[
p = m\frac{dx}{dt} = mv \quad (3.19)
\]

Thus, re-expressing Hamilton’s equations with the Cartesian phase space identities inserted and recognizing that the derivative of the Hamiltonian with respect to position is the same as the derivative of the potential energy \(V\), yields the equations of motion for the system in terms of primitive variables as in equations 3.20 and 3.21.
\[
\frac{d}{dt} \left( m \frac{dx}{dt} \right) = -\frac{\partial H}{\partial x} 
\]
(3.20)

\[
m \frac{d^2 x}{dt^2} = -\frac{\partial V_e}{\partial x} 
\]
(3.21)

In terms more conventional for engineering, the same notion can be expressed in Equation 3.22 with gradient and “dot” notation, as well as considering only a scalar \( x \).

\[
\ddot{x} = -\nabla V_e (x) / m 
\]
(3.22)

The remaining work is to integrate these equations of motion for a particular duration using the previously defined interatomic potentials and any body forces that may act on the system. A very desirable aspect for this integration is conservation of energy, since in equilibrium molecular dynamics the energy of the system should stay fixed in absence of external forces; numerical methods which do not conserve energy would needlessly complicate the statistics of the atomic trajectories. There exists an entire class of integration algorithms called “symplectic” integrators which conserve a perturbed form of the system Hamiltonian through time. Several integration methods commonly used by molecular dynamics software are considered here.

The Verlet family of integrators was first used by Delambre, but has been rediscovered many times by various researchers, including Verlet in the 1960s for use in molecular dynamics. This is likely due to the simplicity of the method, and the ease with with it may be derived using Taylor’s series expansions and finite difference approximations. The method is shown in Equation 3.23.

\[
x (t_{j+1}) = 2x (t_j) - x (t_{j-1}) + \ddot{x} (t_j) \Delta t^2 
\]
(3.23)

One perceived disadvantage of the standard Verlet integrator for use in molecular dynamics is the absence of the velocity in the expression, requiring velocities to be approx-
imated separately. This is resolved by using the Velocity-Verlet algorithm, which can be
derived by making more Taylor’s series and finite difference substitutions. Equations 3.24
and 3.25 show the position and update expressions for the Velocity-Verlet algorithm.

\begin{align*}
x(t_{j+1}) & = x(t_j) + \dot{x}(t_j) \Delta t + \frac{1}{2} \ddot{x}(t_j) \Delta t^2 \\
\dot{x}(t_{j+1}) & = \dot{x}(t_j) + \frac{\Delta t}{2} [\ddot{x}(t_j) + \ddot{x}(t_{j+1})]
\end{align*}

(3.24)  
(3.25)

The method is formally more accurate than first-order Euler’s method with basically
no extra computational cost; it has also proven to be more stable as well. Note that the
position and velocity data are known at the same time step, which presents a convenient
arrangement of variables. In form the integrator is a simple Taylor’s series for the position
update, while the velocity update can be viewed as a first-order Euler method using an
approximation of \( \ddot{x}(t_{j+1/2}) = \frac{1}{2} [\ddot{x}(t_j) + \ddot{x}(t_{j+1})] \), which does not decrease the methods
accuracy any more than the existing approximations.

Another integrator that preserves system energy is the so-called “Leapfrog” method,
which earns the name from the staggered times at which the position and velocity are
known similar the the childrens’ game of the same name. The position and velocity update
expressions are shown in 3.26 and 3.27.

\begin{align*}
x(t_j) & = x(t_{j-1}) + \dot{x}(t_{j-1/2}) \Delta t \\
\dot{x}(t_{j+1/2}) & = \dot{x}(t_{j-1/2}) + \ddot{x}(t_j) \Delta t
\end{align*}

(3.26)  
(3.27)

Derivation of the method only requires a straight-forward application of the central
finite difference approximation of the first derivative to each variable, but staggering the
times at which those variables are stored. Also higher order then Euler’s method, the
Leapfrog method shows stability for oscillatory motion, provided that the timestep is con-
stant and less than half the period of oscillation. This last characteristic makes it especially
attractive for application in molecular dynamics since the simulation of solids is merely
the simulation of a set of chaotic oscillators such as the trajectory of a single arsenic atom
in indium arsenide as seen in Figure 3.10. LAMMPS uses Leapfrog integrators for all its
operation.

3.1.6.1 Thermostatting

LAMMPS implements a Nose-Hoover thermostat for the NPT ensemble, which couples
each atom’s velocity with an external heat bath through modification of the dynamics
equations [97]. These modifications take the following form in equations 3.28 and 3.29.

\[
\ddot{x} = -\nabla V(x) - \eta p \tag{3.28}
\]
\[
\dot{\eta} = \frac{1}{\tau_T} \left( \frac{T}{T_{set}} - 1 \right) \tag{3.29}
\]

The value of \(\tau\) controls how stiffly the system is coupled to the thermal bath; low values
force the system to follow the bath temperature very rapidly and can lead to overly stiff
systems, but large values can lead to very slow convergence to the bath temperature. A \(\tau_P\)
value of about one hundred time steps is recommended for smooth operation [95], which was
confirmed in the early stages of this work. Because the dynamics of the system have been
modified, only the average equilibrium values can be taken as valid; any dynamic property
such as heat transfer must not be computed from such an ensemble. Thus, a thermostat is
only applied while the system is relaxed, not during the simulation to gather data for the
computation of material properties.

3.1.6.2 Barostatting

The barostat implementation in LAMMPS closely follows the same pattern as the
thermostat [95], but uses a separate bath for temperature and pressure as shown in equa-
tions 3.30 and 3.31. The same restrictions on the characteristic time for thermostatting also
apply to the barostat, as well as the limitations on the usefulness of the modified system
dynamics for the computation of material properties.

\[
\begin{align*}
\dot{x} &= -\nabla V(x) - \epsilon p \\
\dot{\epsilon} &= \frac{1}{\tau_p} \left( \frac{T}{T_{set}} - 1 \right)
\end{align*}
\] (3.30) (3.31)

3.1.7 The Brillouin Zone

An important concept in the field of solid state physics is the lattice of a crystalline solid, and all of the aspects that such geometrical symmetry and periodicity implies. The lattice itself is just a geometrical construct which defines a set of points in space such that they repeat using a set of translational transformation vectors, called the lattice vectors. At each site of the lattice, one or more atoms reside, but the addition of any integer multiple of a lattice vector translates the system to an identical configuration of atoms.

Given such periodic arrangement in space, it is only natural to consider the Fourier transform of the structure in space. This leads to the reciprocal lattice, with its reciprocal lattice vectors. If a particular three-dimensional lattice has the lattice vectors \( \mathbf{a}_1, \mathbf{a}_2, \) and \( \mathbf{a}_3 \), then it can be shown that the reciprocal lattice vectors are as found in equations 3.32
From these the lattice can be reconstructed in reciprocal space where many properties are expressed more easily than in real space. Just as the real space lattice has a canonical unit cell, there is a standard unit cell in the reciprocal space called the first Brillouin zone. Figure 3.11 shows the first three Brillouin zones for a particular two-dimensional reciprocal lattice.

The first Brillouin zone is defined by drawing bisectors between the central lattice site and all of its neighbors, and then taking only the innermost volume enclosed by those lines. Each subsequent zone is defined by the number of bisectors crossed in order to arrive at the location starting from the center site.

In the present work the Brillouin zone is most important as the space in which phonon dispersion relations are expressed, so some discussion is needed for the Brillouin zone of the zincblend lattice of III-V semiconductor materials. Figure 3.12a shows one eighth of the Brillouin zone for an FCC lattice, which zincblend structures follow. There are several marks in the figure, which are locations and paths of symmetry in the structure; roman letters

\[
\begin{align*}
    b_1 &= 2\pi \frac{a_2 \times a_3}{a_1 \cdot a_2 \times a_3} \\
    b_2 &= 2\pi \frac{a_3 \times a_1}{a_1 \cdot a_2 \times a_3} \\
    b_3 &= 2\pi \frac{a_1 \times a_2}{a_1 \cdot a_2 \times a_3}
\end{align*}
\]

Fig. 3.11: Brillouin zones for a two-dimensional lattice
denote positions in reciprocal space while greek letters mark special paths between these locations. Because of symmetry from the periodic lattice, a location is actually defined to occupy multiple locations in reciprocal space and there exist multiple paths from one location to another.

In the case of the QCL period considered there are two important directions: in-plane and cross-plane, defined relative to the layers of the cascade. Figure 3.12b shows the FCC first Brillouin zone with distinctions made for these two directions, as well as extending some of the points in to neighboring first Brillouin zones for reasons that are explained later.

3.1.8 Dispersion Relations

Like all waves, phonons possess both a temporal frequency $\omega$ and a spatial wavelength $\lambda$, although the wavelength is frequently represented by a wave number $k = 2\pi/\lambda$. The relationship between the wave number and the frequency is called the dispersion relation, or dispersion curve. For photons, the relationship is linear, since $\lambda f = c$.

For phonons the situation is more complex, and there exists multiple bands which the phonons may occupy. These are broken into the acoustic bands, which are low frequency, and the optical bands, which are high frequency. The optical phonons are so named since they exist in dielectric material and can be excited by electromagnetic fluctuations such as photons. The phonon dispersion curves for a material can be computed from molecular dynamics simulations by processing the velocities and positions of the atoms through time.

The occupation level of a particular phonon state can be computed by taking the spatial and temporal Fourier transforms of the atomic velocities. Since the atoms in the system are discrete, both transforms must be discrete as well. In time, the Fourier transform can be computed using the FFT algorithm, which provides sufficient speed, but the spatial transform must be done directly. This process can be very computationally expensive. Equation 3.35 shows the continuous definition of the phonon mode intensity, while the discrete form is given by Equation 3.36.
The initial implementation of the above algorithm was in Python and read the simulation output as text files. This proved to be too slow, so the output was changed to binary format and the computation was re-written in C, using the FFTW [98] library for the FFT implementation. While much faster, this was still too slow, so the final version was made parallel through OpenMP multithreading [99].

### 3.1.9 Green-Kubo Conductivity Calculation

Two methods are primarily used in molecular dynamics to compute the thermal conductivity of a system. In the non-equilibrium simulation regime, the Muller-Plathe [100] can be used to exchange kinetic energy between two regions of a simulation through swapping the momentum of the fastest and slowest atoms in the regions. This results in a net heat flux between the regions and a temperature gradient forms in the system; the advantage of swapping momentums instead of using thermal baths is that the statistical velocity distributions are left unaffected.
The second method commonly used to compute thermal conductivities is through equilibrium molecular dynamics simulation and the Green-Kubo relation [33,38,101–104]. This formula relies on the heat flux autocorrelation, which is a measure of how a system relaxes back to equilibrium after a fluctuation. The integral of the heat flux autocorrelation is directly tied to the thermal conductivity through the system volume, temperature, and the Boltzmann constant through Equation 3.37.

\[ k = \frac{V}{k_B T^2} \int_0^\infty \langle \mathbf{J}(0) \cdot \mathbf{J}(t) \rangle \, dt \]  

(3.37)

In the present work, only equilibrium simulations were conducted. This is done because the Green-Kubo method is less sensitive to the size of the simulation system, as large systems are more costly to compute. Also the computation of the effective thermal conductivity in a QCL period through non-equilibrium methods is not practical because of the system geometry.

3.2 Photothermal Radiometry

Photothermal methods for property measurements have existed for at least the better-part of a century, although they have seen a dramatic increase in application within the last thirty years; what started with photoacoustic methods relying on microphones has diverged into photothermal radiometry, pyroelectric methods, modulated optical reflectance and the photothermal deflection method, to name only a few. The method of photothermal radiometry (PTR) has attractive features for the measurement of a sample’s thermal properties since the method requires neither contact with or destruction of the sample [93]. This allows sensitive or expensive samples to be economically evaluated, and has made the method attractive for nuclear materials characterization. A typical photothermal radiometry system is show in Figure 3.13; a lock-in amplifier is controlled through computer to run the whole experiment. The amplifier’s internal oscillator is routed to the control input for the laser driver, which results in a continuous wave laser with a intensity modulated at the desired frequency. This heats the surface of the sample, which then re-emits some of
the energy as photons in the infrared range. These photons are focused into an infrared
detector, generally with a mercury-cadmium-telluride (MCT) detector, which generates a
voltage proportional to the received radiation. This is then fed back into the amplifier,
which can then measure the phase lag between the laser and the emitted radiation, which
is a function of the sample’s thermal properties and the frequency of modulation.

There are two key aspects to the experiment. The first of these is the operation of the
lock-in amplifier, which measures the phase difference between two periodic signals. The
second key aspect is the propagation of heat in a periodically excited medium, frequently
termed the thermal wave-field (TWF). Both aspects of the experiment, as well as uncertainty
estimations are considered.

### 3.2.1 Thermal-wave Field Equation

The propagation of heat in a periodically excited system can be analyzed in much the
same way as any transient heat transfer measurement. Excluding boundary conditions, all
heat transfer is assumed to be through conduction, so the transient diffusion heat transfer
equation found in 3.38 can be used.

\[
0 = \alpha \nabla^2 T - \dot{T}
\]  
(3.38)

![Fig. 3.13: Typical photothermal radiometry (PTR) experimental setup](image)
A similar analysis was conducted using the hyperbolic conduction equation, but the frequencies of modulation required for any difference in prediction was well in to the gigahertz range, and thus the investigation was abandoned as unnecessary.

Using the Fourier transform, the frequency-domain heat equation, or thermal-wave field equation can be written directly from the conventional conduction equation. The Fourier transform and variable change are shown in Equation 3.39, while the final thermal wave field equation is given by 3.42 with the thermal wave number as defined in 3.41.

\[
F(T) = \theta, \ t \rightarrow \omega 
\]

\[
0 = F(\alpha \nabla^2 T - \dot{T}) 
\]

\[
q^2 = \frac{i\omega}{\alpha} 
\]

\[
0 = \nabla^2 \theta - q^2 \theta 
\]

All this can be done with no loss of generality from the original conduction equation, but the solution has been moved from the temporal domain to the frequency domain. The thermal-wave field is expressed as a phaser value in much the same way as alternating current by electrical engineers. It can be seen from the form of the thermal-wave field equation that the thermal-wave field at each frequency is independent of all other frequencies; thus only the particular frequency of excitation need be considered.

### 3.2.2 Solutions to the TWF Equation

#### 3.2.2.1 1D Solution

Using Laplace transforms, an analytical solution to the thermal-wave field equation can be readily obtained. First, the general form of the equation must be reduced to the one-dimensional case as shown in 3.43.
Next the Laplace operator is applied (Equation 3.44) and the equation is transformed from real space to Laplace space (equations 3.45 and 3.46).

\[ \mathcal{L}(\theta) = \Theta, \ x \rightarrow \xi \]  
\[ 0 = \mathcal{L}\left( \frac{d^2 \theta}{d{x}^2} - q^2 \theta \right) \]  
\[ 0 = \xi^2 \Theta - \xi \theta_0 - \theta'_0 - q^2 \Theta \]  

Fourier’s law still applies in the transformed coordinates, so the spectral heat flux and thermal-wave derivative can be related as in 3.47.

\[ \theta'_0 = -\frac{1}{k} \phi_0 \]  

Substituting the flux definition back in, and isolating the thermal-wave term, the expression is prepared to transform back into real space (equations 3.48 and 3.49).

\[ 0 = (\xi^2 - q^2) \Theta - \xi \theta_0 + k^{-1} \phi_0 \]  
\[ \Theta = \frac{\theta_0 \xi}{\xi^2 - q^2} - \phi_0 k^{-1} q^{-1} \frac{q}{\xi^2 - q^2} \]  

The transformation into real space yields a surprisingly simple expression shown in 3.50.

\[ \theta = \theta_0 \cosh (qx) - \phi_0 k^{-1} q^{-1} \sinh (qx) \]
From the solution to the thermal-wave field, the flux field can be computed by Fourier’s law as in 3.51.

\[ \phi = -\theta_0 k q \sinh(qx) + \phi_0 \cosh(qx) \]  

(3.51)

These two solutions can then be used to compute the temperature and flux at another location within the domain. This pair of equations is best expressed in the form of matrix operations shown in 3.52.

\[
\begin{bmatrix}
\theta_i \\
\phi_i
\end{bmatrix} =
\begin{bmatrix}
\cosh(qL) & k^{-1}q^{-1} \sinh(qL) \\
kq \sinh(qL) & \cosh(qL)
\end{bmatrix}
\begin{bmatrix}
\theta_{i+1} \\
\phi_{i+1}
\end{bmatrix}
\]  

(3.52)

The relation in equations 3.53 through 3.56 can be written between two surfaces within a material.

\[
\begin{bmatrix}
\theta_i \\
\phi_i
\end{bmatrix} =
\begin{bmatrix}
K_i & F_i \\
A_i & K_i
\end{bmatrix}
\begin{bmatrix}
\theta_{i+1} \\
\phi_{i+1}
\end{bmatrix}
\]  

(3.53)

\[ K_i = \cosh(q_i L_i) \]  

(3.54)

\[ F_i = (k_i q_i)^{-1} \sinh(q_i L_i) \]  

(3.55)

\[ A_i = (k_i q_i) \sinh(q_i L_i) \]  

(3.56)

This can be generalized to layered samples with front \((f)\) and back \((b)\) surfaces, as well as inner \((i)\) layers as in Equation 3.57. This method of calculation is called the quadrupole method, and represents each layer within the sample by an impedance matrix.
\[
\begin{bmatrix}
\theta_f \\
\phi_f 
\end{bmatrix} = \prod_i \begin{bmatrix}
K_i & F_i \\
A_i & K_i 
\end{bmatrix} \begin{bmatrix}
\theta_b \\
\phi_b 
\end{bmatrix} 
\] (3.57)

With an insulated back and known face surface flux, the surface temperature can be computed, as shown in equations 3.58 through 3.60.

\[
\begin{bmatrix}
a & b \\
c & d 
\end{bmatrix} = \prod_i \begin{bmatrix}
K_i & F_i \\
A_i & K_i 
\end{bmatrix} 
\] (3.58)

\[
\begin{bmatrix}
\theta_f \\
\phi_f 
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d 
\end{bmatrix} \begin{bmatrix}
\theta_b \\
0 
\end{bmatrix} 
\] (3.59)

\[
\theta_f = \frac{a}{c} \phi_f 
\] (3.60)

A simple resistive layer can be modeled by forcing the layer thickness to zero while keeping the total resistance constant (Equation 3.61).

\[
\lim_{L \to 0} \begin{bmatrix}
K_i & F_i \\
A_i & K_i 
\end{bmatrix} = \begin{bmatrix}
1 & R_i \\
0 & 1 
\end{bmatrix} 
\] (3.61)

3.2.2.2 1D Analytical Solution Stabilization

The quadrupole method can be used to solve the thermal-wave field equation for layered samples. After considering the sensitivity of the computed phase to the input thermal properties, it is noticed that the thermal diffusivity \( \alpha \) and thermal conductivity \( k \) are non-orthogonal to one another. This is easily explained by considering the term \( k_iq_i \) which appears in the off-diagonal terms of the impedance matrix. Upon examination, this quantity can be expressed more correctly in terms of thermal effusivity \( \epsilon = \sqrt{kC} \), where \( C \) is the
Fig. 3.14: Example solutions of the thermal-wave field equation in space and the surface solution verses frequency: a) thermal-wave field in space showing both wave intensity and phase, including a discontinuity from contact resistance; b) surface wave intensity and phase verses frequency

Volumetric specific heat.

Using this definition, the elements in the impedance matrix can be re-expressed without the interdependence as follows in equations 3.62 through 3.67:

\[
K_i = \cosh(q_i L_i) \tag{3.62}
\]
\[
E_i = \sinh(q_i L_i) \tag{3.63}
\]
\[
F_i = p_i^{-1} E_i \tag{3.64}
\]
\[
A_i = p_i^{+1} E_i \tag{3.65}
\]
\[
q_i = \sqrt{\frac{i \omega}{\alpha_i}} \tag{3.66}
\]
\[
p_i = \epsilon_i \sqrt{i \omega} \tag{3.67}
\]

While this form of the equations is an improvement over the previous form, it still suffers from a serious problem: if the quantity \( q_i L_i \) is large then it causes floating-point exceptions during evaluation of the hyperbolic functions. On some computers this can occur for values
as small as 700 \((1 + i)\). This obviously prevents the method from being applied to samples with thick layers (large \(L_i\)) or very low diffusivities (large \(q_i\)). Fortunately, this problem can be resolved.

Firstly, the thermal diffusion length (Equation 3.69) is introduced to simplify the definition of \(q\) in equations 3.68 and 3.70.

\[
qL = \sqrt{\frac{i\omega}{\alpha}} L \quad (3.68)
\]

\[
\mu = \sqrt{\frac{2\omega}{\alpha}} \quad (3.69)
\]

\[
qL = (1 + i) \frac{L}{\mu} \quad (3.70)
\]

The hyperbolic functions can then be expanded using sum and imaginary argument identities as in 3.71 and 3.72.

\[
K = \cosh \left( \frac{L}{\mu} \right) \cos \left( \frac{L}{\mu} \right) + i \sinh \left( \frac{L}{\mu} \right) \sin \left( \frac{L}{\mu} \right) \quad (3.71)
\]

\[
E = \sinh \left( \frac{L}{\mu} \right) \cos \left( \frac{L}{\mu} \right) + i \cosh \left( \frac{L}{\mu} \right) \sin \left( \frac{L}{\mu} \right) \quad (3.72)
\]

New coefficients are then defined through normalization by \(\cosh (L/\mu)\) (equations 3.73 and 3.74).
\[ K' = \frac{K}{\cosh(L/\mu)} \]
\[ = \cos \left( \frac{L}{\mu} \right) + i \sin \left( \frac{L}{\mu} \right) \tanh \left( \frac{L}{\mu} \right) \]  
\[ (3.73) \]

\[ E' = \frac{E}{\cosh(L/\mu)} \]
\[ = \tanh \left( \frac{L}{\mu} \right) \cos \left( \frac{L}{\mu} \right) + i \sin \left( \frac{L}{\mu} \right) \]  
\[ (3.74) \]

These new coefficients are then used to define the modified impedance matrix $3.76$ from $3.75$.

\[ [Z] = \begin{bmatrix} K & E/p \\ E p & K \end{bmatrix} \]  
\[ (3.75) \]

\[ [Z'] = \frac{[Z]}{\cosh(L/\mu)} \]
\[ = \begin{bmatrix} K' & E'/p \\ E' p & K' \end{bmatrix} \]  
\[ (3.76) \]

The product of the standard impedance matrices can be expressed as the product of the modified matrices and the product of the normalizing factors as in equations $3.77$ through $3.79$.

\[ \prod_i [Z_i] = \prod_i \left\{ \cosh \left( \frac{L_i}{\mu_i} \right) [Z'_i] \right\} \]  
\[ = S^{-1} \prod_i [Z'_i] \]  
\[ S^{-1} = \prod_i \cosh \left( \frac{L_i}{\mu_i} \right) \]  
\[ (3.77) \]

\[ (3.78) \]

\[ (3.79) \]
This form is substituted into the relation between the front and back faces (equations 3.80 and 3.81).

\[
\begin{bmatrix}
\theta_f \\
\phi_f
\end{bmatrix} = \prod_i [Z_i] \begin{bmatrix}
\theta_b \\
\phi_b
\end{bmatrix} = S^{-1} \prod_i [Z'_i] \begin{bmatrix}
\theta_b \\
\phi_b
\end{bmatrix}
\]  

(3.80)

(3.81)

Upon solving for the surface temperature with a known surface flux and insulated back face, the normalizing factors cancel out, completely removing them from the computation. This remarkable result from the operations in equations 3.82 through 3.86.

\[
\prod_i [Z'_i] = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
\]  

(3.82)

\[
\begin{bmatrix}
S\theta_f \\
S\phi_f
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix} \begin{bmatrix}
\theta_b \\
\phi_b
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix} \begin{bmatrix}
\theta_b \\
0
\end{bmatrix}
\]  

(3.83)

(3.84)

\[
\theta_b = \frac{S\phi_f}{c}, \quad S\theta_f = S\frac{a}{c}\phi_f
\]  

(3.85)

\[
\theta_f = \frac{a}{c}\phi_f
\]  

(3.86)

By normalizing the impedance matrix, evaluations of \(\cosh\) and \(\sinh\) are replaced with evaluations of \(\tanh\). This prevents terms in the impedance matrix from becoming exponentially large, although a thorough study of the floating point error in both cases is still to be done.
3.2.2.3 2D Numerical Solution

The axisymmetric form of the thermal-wave field equation can be solved numerically without any undue difficulty. Figure 3.15 shows such a solution for a substrate coated by a thin film. The plot shows the scaled amplitude of the thermal-wave intensity as a function of position within the sample, based on a complex-domain solution for the thermal-wave intensity and phase. The interface between the two materials can be clearly seen in the form of a slope discontinuity, and it can be seen that the waves travel less rapidly through the film than the substrate since the surface wave propagation lags behind that of the substrate; this indicates that the film thermal diffusivity is less than that of the substrate.

The solution was obtained by a finite-volume discretization in cylindrical coordinates and assumed axisymmetry, custom-developed for this work. The complex form of the equation was used, and several linear solvers were tested on this class of problem. Gauss-Seidel proved effective at solving for the thermal-wave intensity, but the phase in regions of very low temperature fluctuation would not converge in an acceptable time frame because of the very small variations in solution variables which drive them. Stone’s strongly implicit scheme was also tested, which not only converged in regions of low wave intensity, did so at a much faster rate; the improvement in convergence rate over Gauss-Seidel exceeds that seen when comparing the two methods on the simple diffusion equation. Finally, a banded direct solver was tested which provides adequate performance and guaranteed accuracy in the phase, so this solver was used for all subsequent simulations.

Fig. 3.15: Example numerical solution of the axisymmetric thermal-wave field equation
3.2.2.4 2D Analytical Solution

Starting by reducing the problem to the two-dimensional cylindrical case, the problem can be written as in Equation 3.87.

\[
0 = \frac{\partial^2 \theta}{\partial r^2} + \frac{1}{r} \frac{\partial \theta}{\partial r} + \frac{\partial^2 \theta}{\partial z^2} - q^2 \theta \tag{3.87}
\]

By performing the Hankel transform as in equations 3.88 through 3.90, the \(r\)-terms are eliminated.

\[
\mathcal{H}(\theta) = \Theta, \ r \to \delta \tag{3.88}
\]

\[
0 = \mathcal{H} \left[ \frac{\partial^2 \theta}{\partial r^2} + \frac{1}{r} \frac{\partial \theta}{\partial r} + \frac{\partial^2 \theta}{\partial z^2} - q^2 \theta \right] \tag{3.89}
\]

\[
0 = \frac{\partial^2 \Theta}{\partial z^2} - (q^2 + \delta^2) \Theta \tag{3.90}
\]

\[
\beta^2 = q^2 + \delta^2 \tag{3.91}
\]

\[
0 = \frac{\partial^2 \Theta}{\partial z^2} - \beta^2 \Theta \tag{3.92}
\]

The problem can be solved using Laplace transforms just as in the one-dimensional case as in equations 3.93 through 3.99.

\[
\mathcal{L}[\Theta] = \Lambda, \ z \to \zeta \tag{3.93}
\]

\[
0 = \mathcal{L} \left[ \frac{\partial^2 \Theta}{\partial z^2} - \beta^2 \Theta \right] \tag{3.94}
\]

\[
0 = \zeta^2 \Lambda - \zeta \Theta_0 - \Theta'_0 - \beta^2 \Lambda \tag{3.95}
\]

\[
\Theta'_0 = -\frac{1}{k} \Phi_0 \tag{3.96}
\]

\[
0 = (\zeta^2 - \beta^2) \Lambda - \zeta \Theta_0 + k^{-1} \Phi_0 \tag{3.97}
\]

\[
\Lambda = \Theta \frac{\zeta}{\zeta^2 - \beta^2} - \Phi_0 k^{-1} \beta^{-1} \frac{\beta}{\zeta^2 - \beta^2} \tag{3.98}
\]

\[
\Theta = \Theta_0 \cosh (\beta z) - \Phi_0 k^{-1} \beta^{-1} \sinh (\beta z) \tag{3.99}
\]
From the solution to the temperature field, the flux field can be computed by Fourier’s law in Equation 3.100.

\[ \Phi = -\Theta_0 k_\beta \sinh (\beta z) + \Phi_0 \cosh (\beta z) \] (3.100)

These two solutions can be used to compute the temperature and flux at another location within the domain. The relation in 3.101 through 3.105 can be written for two surfaces of a material.

\[
\begin{bmatrix}
\Theta_i \\
\Phi_i
\end{bmatrix} =
\begin{bmatrix}
U_i & E_i/p_i \\
E_i \cdot p_i & U_i
\end{bmatrix}
\begin{bmatrix}
\Theta_{i+1} \\
\Phi_{i+1}
\end{bmatrix}
\]

(3.101)

\[ U_i = \cosh (\lambda_i) \] (3.102)

\[ E_i = \sinh (\lambda_i) \] (3.103)

\[ p_i = k_i \beta_i \] (3.104)

\[ \lambda_i = \beta_i L_i \] (3.105)

This can be generalized to layered samples with front \((f)\) and back \((b)\) surfaces, as well as inner \((i)\) layers (Equation 3.106).

\[
\begin{bmatrix}
\Theta_f \\
\Phi_f
\end{bmatrix} = \prod_i \begin{bmatrix}
U_i & E_i/p_i \\
E_i \cdot p_i & U_i
\end{bmatrix}
\begin{bmatrix}
\Theta_b \\
\Phi_b
\end{bmatrix}
\]

(3.106)

With an insulated back and known face surface flux, the surface temperature can be computed as in equations 3.107 through 3.109.
\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix} = \prod_i \begin{bmatrix}
U_i & E_i/p_i \\
E_i \cdot p_i & U_i
\end{bmatrix}
\]  
(3.107)

\[
\begin{bmatrix}
\Theta_f \\
\Phi_f
\end{bmatrix} = \begin{bmatrix}
A & B \\
C & D
\end{bmatrix} \begin{bmatrix}
\Theta_b \\
0
\end{bmatrix}
\]  
(3.108)

\[
\Theta_f = \frac{A}{C} \Phi_f
\]  
(3.109)

Recall that \(\Theta\) and \(\Phi\) are in Hankel space and must be transformed before use. Since the flux profile and phase are assumed to be known, they can be transformed into Hankel space. Assume a Gaussian surface flux profile as shown in Equation 3.110 and 3.111.

\[
\phi_f(r) = I_0 \exp \left( -r^2/a^2 \right)
\]  
(3.110)

\[
\Phi_f(\delta) = I_0 \frac{a^2}{2} \exp \left( -\delta^2 a^2 / 4 \right)
\]  
(3.111)

The inverse Hankel transform is then used to compute the final thermal wave field. While this expression (from equations 3.112 and 3.113) is still in integral form, the integration is not difficult to implement using adaptive algorithms with error control.

\[
\Theta_f(\delta, \omega) = \frac{A(\delta, \omega)}{C(\delta, \omega)} \Phi_f
\]

\[
= I_0 \frac{a^2}{2} \frac{A(\delta, \omega)}{C(\delta, \omega)} \exp \left( -\delta^2 a^2 / 4 \right)
\]  
(3.112)

\[
\theta_f(r, \omega) = \int_0^\infty \Theta_f(\delta, \omega) J_0(\delta r) \delta d\delta
\]

\[
= I_0 \frac{a^2}{2} \int_0^\infty \frac{A(\delta, \omega)}{C(\delta, \omega)} \exp \left( -\delta^2 a^2 / 4 \right) J_0(\delta r) \delta d\delta
\]  
(3.113)

### 3.2.3 Lock-in Amplifier Operation

A lock-in amplifier can isolate a signal from noise which would otherwise dominate
a measurement by extracting the signal at a particular frequency of interest and filtering out all other frequencies. The operation of a lock-in amplifier is much like a hardware implementation of the Fourier transform in time as in Equation 3.114. An input signal is multiplied by an orthogonal pair of reference signals at a particular frequency and the results integrated over a particular time to produce the orthogonal components of the signal at that frequency as shown in Equation 3.115. These components can be used to compute the amplitude (Equation 3.116) and phase (Equation 3.117) of the input signal at the specified frequency.

\[
L (t) = \int_{t-\tau}^{t} S (t) \exp (i\omega t) d\tau \quad (3.114)
\]

\[
A (t) + iB (t) = \int_{t-\tau}^{t} S (t) \sin (\omega t) d\tau + i \int_{t-\tau}^{t} S (t) \cos (\omega t) d\tau \quad (3.115)
\]

\[
R (t) = \sqrt{A^2 (t) + B^2 (t)} \quad (3.116)
\]

\[
\phi (t) = \arctan \left( \frac{B (t)}{A (t)} \right) \quad (3.117)
\]

The selection of an appropriate integration time constant \( \tau \) is critical for accurate measurement of the amplitude and phase of a signal. If the time constant is chosen to be too large, transients in the signal are not recognized in the frequency domain and measurements take too long, but if the time constant is too small then noise overwhelms the actual signal.

3.2.4 Curve-fitting Procedures

As can be seen from the various solutions to the thermal-wave field equation, to measure the thermal properties of a sample that sample must be layered; otherwise, the phase response of the system remains at 45° independently from the frequency. If there are layers with different thermal properties, however, the phase depends on frequency and the phase versus frequency plot can be fit with an appropriate model to recover the sample’s thermal properties. The phase versus frequency data is most often called a frequency scan, and experiments are almost always arranged to guarantee that the one-dimensional solution
applied to the whole scan range.

Once the frequency scan data has been taken, there remains the problem of how to fit it. Many approaches have been proposed for this process, which range from directly modifying the sample properties guess until they match the scan to complex multi-tiered least-squares fitting procedures. In any case, the objective of the curve-fitting process is to minimize some residual function which represents the difference between the measured scan and a candidate property set. In the case of manually fitted data, this residual function is evaluated visually and subjectively; this allows for expertise and acumen to lead the fitting process. In automated fitting procedures, the residual function is almost always a least-squares type difference between the measured scan and the candidate results.

For the best sensitivity and most convenience while fitting, the material properties of the sample are expressed in terms of the thermal diffusivity and thermal effusivity. An example plot of a least-squares type residual field as a function of these properties is seen in Figure 3.16. The point of minimum residual is marked on the plot, and labeled as well for clarity. The data used to generate the figure assumes only two layers in the sample, a thin film over a substrate of known properties, leaving only two unknown properties to fit; if more layers are unknown the method can still work, but the residual function becomes higher-dimensional and plotting becomes more difficult.

A new fitting method is currently being developed which may merge the more attractive aspects of manual fitting with the automation of least-squares fitting. This can be accomplished by creating a set of residual functions: one for each degree of freedom in the fit. Each residual is defined as the weighted sum of least-square differences where the weight function is the normalized sensitivity of the candidate solution to the considered degree of freedom. The set of residuals are then minimized as a group. This should allow the variables to better-drive the fitting process in regions where they make the most difference. The successfulness of this method is reported later.
3.2.5 Monte Carlo Uncertainty Analysis

To evaluate the uncertainty incident to photothermal measurements of material properties, Monte Carlo uncertainty analysis has been conducted. Traditional Taylor’s series uncertainty propagation cannot work on the curve fitting process used in photothermal material properties measurement, due to the fitting process.

Shown in Figure 3.17, the Monte Carlo uncertainty propagation method works by assuming statistical distributions for the actual properties of a sample and then simulating the data acquisition process. Using a random population for the inputs of the simulation results in an output distribution. Each frequency scan in the output distribution is then fit as usual, resulting in a final distribution of material properties. The statistical relationship between the initial property distribution and the final calculated distribution yields the bias and precision of the measurement, provided that the simulation is of sufficient fidelity.

3.3 Tasks to Complete Objectives

In order to accomplish the aforementioned objectives, several tasks have been identified. These are the following:

1. Tersoff coefficients from the literature must be encoded for consumption by LAMMPS during simulations. This has been accomplished by a Python script which automati-
cally expands the pair potentials found in the literature to the triplet potential form required by LAMMPS.

2. LAMMPS simulations have been designed which compute various properties for the validation of literature coefficients.

(a) The thermal expansion coefficient can be computed from variation of the lattice parameter with change in temperature. By running simulations at various temperatures in NTP, the temperature-dependent lattice parameter can be found after the system has been relaxed. The thermal expansion coefficient is then the normalized derivative of lattice parameter with regard to temperature.

(b) Using the same set of simulations, the system energy can be monitored as a function of temperature. The derivative of the energy with regard to the temperature is proportional to the asymptotic high-temperature specific heat. Since the thermostat adds energy until the average kinetic energy of the atoms matches the desired temperature, the thermostat can be used to vary the energy with temperature in a physically significant manner.

(c) The dispersion relations for each material combination in the QCL has been computed by simulation of the system in NVE while recording the atomic positions
and velocities at an appropriate interval. The data must be stored in binary for both performance and storage size considerations. By taking the Fourier transforms of the atomic velocities in both space and time, the dispersion relation is revealed as peak resonances. The polarization of the velocity component considered allow the various branches (transverse and longitudinal) to be discerned. Dispersion calculations must be done in parallel in C for performance reasons.

(d) The thermal conductivity of each composite material can be calculated using several features of LAMMPS combined with post-processing done in Python. The system heat flux must be computed, which is a function of the atomic kinetic energy and stress tensor. While LAMMPS can compute the system heat flux, the flux autocorrelation and its integral has been computed using Python as a post-processing step. Because of the need for ensemble averaging, many simulations are run, and the flux autocorrelations from each simulation are averaged and integrated. From this integral the thermal conductivity can be computed from the Green-Kubo relation. No comparison with the non-equilibrium calculations has been made.

3. To construct the photothermal radiometry, several steps must be taken:

(a) The components for the PTR system must be selected, purchased and tested.

(b) Once acquired and tested, the components must be assembled into the PTR system and the optical components aligned.

(c) A computer program must be written to control the lock-in amplifier, which in turns controls the other components of the experiment.
Chapter 4

Results

4.1 Validation of Potential

Before the results of simulating a quantum cascade laser diode period can be considered, it must first be established that the inter-atomic potential used for the simulations can accurately reproduce the thermal properties of the materials in question. To this end, a series of validation simulations were run with the potential coefficients found in the literature to provide evidence of what thermal properties they predict for the various III-V semiconductors used in QCL composition.

Using the Tersoff potential and coefficients listed in table 3.2, the thermal expansion coefficient, specific heat, phonon density of states, phonon dispersion, thermal conductivity and spectral conduction have been computed for the compounds and alloys used in the QCL under consideration. A review of the results from these simulations including comparisons with experimental values when available immediately follows.

4.1.1 Thermal Expansion

The thermal expansion coefficient relates the change in size of a material with changes in temperature, and is a commonly encountered phenomenon in engineering endeavors which is routinely accounted for. Like many every-day occurrences, this material behavior results from the fundamental atomic interactions within the material, specifically the tendency of the atoms to move slightly further apart at higher temperatures. This can be easily understood when it is remembered that the temperature is a function of the average kinetic energy of the atoms in a locale, and the faster atoms cause greater reaction forces between them, thereby pushing them apart.
There are two coefficients commonly used to describe thermal expansion, the linear coefficient found in Equation 4.1, and the volumetric coefficient in Equation 4.2. The first version considers only the expansion in a single dimension, while the second considers the volume of a body as a whole. Obviously in the case of anisotropic expansion, a single material may exhibit a thermal expansion tensor instead of a single coefficient.

\[
\alpha_L = \frac{1}{a} \frac{da}{dT} \tag{4.1}
\]

\[
\alpha_V = \frac{1}{V} \frac{dV}{dT} \tag{4.2}
\]

It can be demonstrated without difficulty that in the case of an isotropic material the two coefficients are related by a factor of three, as shown in Equation 4.3.

\[
\alpha_V = 3\alpha_L \tag{4.3}
\]

To compute the thermal expansion coefficients predicted by the present potential, a small system of atoms is constructed from each of the materials of interest. The size is relatively unimportant, and increasing the number of atoms considered does not appreciably affect the resulting coefficient. In these simulations, the system was a cube with an edge length of four unit cells, resulting in a system composed of 512 atoms. While the scripts used to build the system normally randomly select which atom occupies a lattice site for ternary systems, this was found to cause trouble in the thermal expansion simulations, since the changes to the system composition would cause larger shifts in energy than the temperature variation. Thus, a fixed seed value is given to the setup script for these simulations to ensure that the same system is used at all temperatures.

This system is then simulated for ten picoseconds using NTP dynamics (constant atoms, temperature, and pressure), allowing the system size to adjust to the potential. After this, the system is again simulated for another ten picoseconds, over which time the lattice
parameter is averaged along with the actual temperature, the kinetic energy, and potential energy. The simulation was repeated at a variety of temperatures between 100 K and 500 K, with each simulations’ results tabulated in a file. The lattice parameter versus temperature data was fit to a quadratic polynomial, and then that polynomial was substituted into Equation 4.1 to compute the linear thermal expansion coefficient for the material across the temperatures. A quadratic polynomial was selected simply because it seemed to best match the shape of the data from the simulations, but was still of lower order to prevent spurious oscillations in the fit.

The results of each simulation can be found in Figures 4.1 through 4.6, along with experimental results for each material when it could be found in the literature. It can be seen that with the potential used for the simulations the lattice parameter is a smooth function of temperature, and that the quadratic polynomial selected for the curve fit can adequately represent its variation. Even though the form of Equation 4.1 after the quadratic polynomial is substituted suggests that the thermal expansion coefficient should behave as a rational function of a linear polynomial over a quadratic, it can be seen from the plots of lattice parameter versus temperature that the variation within the temperature range is minuscule, effectively resulting in a constant divided by a linear function.

From the plots, it can be seen that when experimental data is available, the simulation results predict a thermal expansion coefficient of roughly double the experimental value. While this is certainly undesirable, it is not unexpected, since the potential coefficients are rarely fit using any thermal properties at all, and the ability to even partially recover thermal properties from coefficients developed using only mechanical properties is actually quite an impressive accomplishment of molecular dynamics. Even if the thermal expansion was considered when the coefficients were fit (which is unlikely), the fitting process is always a balance between the various properties used to compare simulation results with experiments; depending on how well the potential function matches the real inter-atomic interactions, it may be impossible to reproduce all observed material properties with a particular function.

It can also be seen from the plots for AlAs and AlGaAs that there is little information
Fig. 4.1: Thermal expansion results for AlAs

Fig. 4.2: Thermal expansion results for GaAs

Fig. 4.3: Thermal expansion results for InAs
Fig. 4.4: Thermal expansion results for AlGaAs

(a) Lattice constant vs temperature  
(b) Linear thermal expansion coefficient [105]

Fig. 4.5: Thermal expansion results for AlInAs

(a) Lattice constant vs temperature  
(b) Linear thermal expansion coefficient

Fig. 4.6: Thermal expansion results for GaInAs

(a) Lattice constant vs temperature  
(b) Linear thermal expansion coefficient
about the thermal expansion of the materials, and that between experimental values there is some disagreement. If the mean value of the literature data is taken, the simulations are again close to twice the value from experiments. The presence and absence of experimental data here demonstrates a pattern found in the literature as this work was conducted; among the arsenide compounds, GaAs and InAs are the most thoroughly characterized, followed by AlAs, and then many ternary systems can not be found in the literature at all for thermal properties. It is supposed that many of these properties have been measured, but are considered trade secrets by the companies that have used these materials in various semiconductor products. The results from Figures 4.1 through 4.6 are summarized in table 4.1; again when there are multiple values for a material, the table reports only the mean of those values.

In the absence of experimental data, but in light of the obvious trend seen among the materials, it is expected that the true thermal expansion coefficients for AlInAs and GaInAs are approximately half of the values predicted by these simulations. While there is no direct proof of this claim, it is reasonable and demonstrates an important use of molecular dynamics: the capacity to reasonably predict material properties for substances which have not been, and possibly cannot be measured.

4.1.2 Specific Heat

Using the same set of simulation data collected to compute the thermal expansion coefficient, the specific heat of the materials can be computed from the change in system

<table>
<thead>
<tr>
<th>III-V Material</th>
<th>Experimental $\alpha_L [K^{-1}]$</th>
<th>Computed $\alpha_L [K^{-1}]$</th>
<th>Difference [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlAs [105]</td>
<td>$4.3 \times 10^{-6}$</td>
<td>$9.3 \times 10^{-6}$</td>
<td>+116</td>
</tr>
<tr>
<td>GaAs [106]</td>
<td>$5.8 \times 10^{-6}$</td>
<td>$11.9 \times 10^{-6}$</td>
<td>+105</td>
</tr>
<tr>
<td>InAs [107]</td>
<td>$4.4 \times 10^{-6}$</td>
<td>$9.9 \times 10^{-6}$</td>
<td>+125</td>
</tr>
<tr>
<td>AlGaAs [105]</td>
<td>$5.4 \times 10^{-6}$</td>
<td>$10.4 \times 10^{-6}$</td>
<td>+93</td>
</tr>
<tr>
<td>AlInAs</td>
<td>-</td>
<td>$10.3 \times 10^{-6}$</td>
<td>-</td>
</tr>
<tr>
<td>GaInAs</td>
<td>-</td>
<td>$13.4 \times 10^{-6}$</td>
<td>-</td>
</tr>
</tbody>
</table>
energy versus temperature. From thermodynamics it can be recalled that the specific heat capacity can be defined as the change in energy of a system with respect to temperature, and that there are two varieties of the property. If the system volume is held constant as energy is added or removed, the specific heat is labeled $C_V$, while if the pressure is held constant, the specific heat is labeled $C_P$. For solids, the two properties are practically identical, but for accuracy’s sake it should be noted that the results here are technically $C_P$, since it is the pressure which was held (nearly) constant during simulation. Thus, the property can be defined as in Equation 4.4, where $E$ is the total system energy and $M$ is the total system mass.

$$C_P = \frac{1}{M} \left( \frac{\partial E}{\partial T} \right)_P$$  \hspace{1cm} (4.4)

At the nanoscale, the specific heat is a direct measurement of the degrees of freedom of the system, and real systems demonstrate many complex effects from quantum mechanics which result in specific heat values that approach zero as the temperature reaches zero Kelvin and follow an approximate $C_P \propto T^3$ law. Because molecular dynamics does not include any quantum mechanical effects directly, none of these behaviors are expected in the specific heat results from the simulations, but rather the results are expected to follow the asymptotic high-temperature specific heats of the materials.

To compute the specific heat from the thermal expansion simulations, the system energy as a function of temperature is fit to a quadratic polynomial; this polynomial is then substituted into Equation 4.4 to compute the specific heat, which is assumed to be linear. While molecular dynamics should only be able to recover the high-temperature specific heat (which is constant) allowing the resulting form to be linear provides a chance to assess how well that condition is met. Figures 4.7 through 4.12 show the results of the simulation as well as the computed specific heats as compared with experimental values for the materials at room temperature.

As can be seen from the plots, the results for the specific heat are much closer to
Fig. 4.7: Specific heat results for AlAs

Fig. 4.8: Specific heat results for GaAs

Fig. 4.9: Specific heat results for InAs
Fig. 4.10: Specific heat results for AlGaAs

Fig. 4.11: Specific heat results for AlInAs

Fig. 4.12: Specific heat results for GaInAs
experimental values than what was obtained for the thermal expansion coefficients. Since the specific heat is a fundamental, but simple, property, this is to be expected. Also of note is that the variation across the range of temperatures is not very large, and is actually always increasing (which matches experimental results not shown). The results from the simulations for specific heat are summarized in table 4.2.

From the differences observed in the table, it is clear that the expected error in specific heat predictions by simulations using the current potential is on the order of ten percent. It must be noted, also, that the “experimental” result for AlInAs in the table is actually an average of the values from AlAs and InAs, which is in following with models of ternary systems found in the literature. The expected error from the approximation is on the same order as that of the simulations.

4.1.3 Dispersion Curves and the Density of States

The dispersion curves and density of states are both used to indicate which phonon modes are occupied in a given system, and can provide qualitative information about how the system is expected to behave. Their definition is fairly straightforward, but computing them from the results of molecular dynamics simulations turned out to be quite complex, especially in the case of the dispersion curves. Partly because of this complexity but also due to the somewhat excessive computing requirements for the dispersion relation, specialized software had to be written which dramatically improved the performance when computing the density of states and dispersion relations by using binary I/O and parallel processing.

<table>
<thead>
<tr>
<th>III-V Material</th>
<th>Experimental ( C_p [J/kg \cdot K] )</th>
<th>Computed ( C_p [J/kg \cdot K] )</th>
<th>Difference [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlAs</td>
<td>[108]450</td>
<td>508.2</td>
<td>+13</td>
</tr>
<tr>
<td>GaAs</td>
<td>[109]330</td>
<td>367.4</td>
<td>+11</td>
</tr>
<tr>
<td>InAs</td>
<td>[110]250</td>
<td>279.2</td>
<td>+12</td>
</tr>
<tr>
<td>AlGaAs</td>
<td>[108, 109]390</td>
<td>430.3</td>
<td>+11</td>
</tr>
<tr>
<td>AlInAs</td>
<td>[108, 110]350</td>
<td>366.0</td>
<td>+2</td>
</tr>
<tr>
<td>GaInAs</td>
<td>[111]300</td>
<td>321.5</td>
<td>+7</td>
</tr>
</tbody>
</table>
4.1.3.1 Density of States Calculation

The phonon density of states (DOS), which is often referred to as simply the density of states, is a function which expresses the number of phonon modes in a system that are occupied at a particular frequency. There are many applications of the density of states that is not be contemplated here, but it serves as a starting point in computing the dispersion curves, and can also be used for potential validation; no literature values were found that were suitable for final comparison, but some were referenced in the early stages of the present work to serve as initial sanity checks of the potential.

Most commonly used for the computation of the density of states is a relaxation form, given in Equation 4.5. While this expression undoubtedly works, it must be discretized and there is no obvious way to expand the definition to an entire system of atoms directly from the expression.

\[
\text{DOS} (\omega) = \int_0^\infty \frac{\langle v(t) \cdot v(0) \rangle}{\langle v(0) \cdot v(0) \rangle} \exp (-i\omega t) \, dt \tag{4.5}
\]

Because of this, an alternative form is used to compute the density of states, given in Equation 4.6. It can be seen immediately from the definition of the function that it requires accessing the velocity of each atom at each relevant time step, and that in general the processing for each atom is independent from the others. These characteristics lead to an algorithm that can take advantage of modern multi-core CPUs to dramatically speed up the process. Since the velocity at all time steps is needed for each atom’s Fourier transform, the data first had to be converted into a format that makes this possible; the netCDF format was selected due to its simplicity and effectiveness.

\[
\text{DOS} \{\omega\} \approx \sum_{k \in \text{atoms}} \left[ \sum_{i \in \{x,y,z\}} |\text{FFT}_j \left[ v_k(t_j) \cdot n_i \right] |^2 \right] \tag{4.6}
\]

By default, most output from LAMMPS is in an ASCII-based format, which lists data
for each time step as a table with a row for each atom and one column for each output property. The atoms are not sorted in the output process, since that would unacceptably slow down the writing process. Each time step block is separated by a header which allows them to be sorted out later by post-processing software. In this work, the density of states for small systems was computed from the ASCII output using Python scripts which read the entire simulation result file into computer memory. On larger simulation systems this is not possible, since the quantity of data which must be processed can exceed tens of gigabytes, and while there exist computers in USU’s HPC resources with sufficient memory to store such results, a better solution was pursued.

Firstly, the simulations scripts for LAMMPS were changed to write a binary-based output file in place of the ASCII format, which reduces the size of files by about a factor of two, but also improve the write performance during the simulation since the in-memory number representation is just transfered to disk instead of being first converted to an ASCII representation of the number. A series of functions were written in the C programming language which can read the LAMMPS binary format, based initially on a C++ reader program available on the LAMMPS website. These functions were constructed so that a single time step could be read from the binary file at a time, and the atoms within that step sorted according to their ID tag. Each time step is then sequentially written to a NetCDF file, which is a standard scientific data format which provides convenient but performant I/O routines to access data [112, 113]. The simulation results are represented by a three-dimensional array within the NetCFD file, which can be accessed using array slices just as in Fortran/90+ or Matlab®.

With the data in a more useful data format, it was then much less difficult to process each atom separately, as a single atom’s data for all time steps could be read from the file and processed directly. The processing of the atoms was parallelized using OpenMP so that all the cores of the computer could be used [99]. It has been found from experience with processing the results that the performance of this algorithm is generally bound by disk I/O performance, and the CPU cores are not fully utilized. While this could probably be
improved, the present performance is sufficient and the computing time for the density of states is still dwarfed by that for the dispersion relations.

### 4.1.3.2 Dispersion Calculation

The most commonly used method of producing phonon dispersion curves with molecular dynamics is to compute the dynamical structure factor (Equation 4.7) and to look for ridges in the resulting surface [114]. The dynamical structure factor is defined for a particular wave vector $\mathbf{k}$ and frequency $\omega$, but requires atomic positions $\mathbf{x}$ for its actual computation.

$$S(\omega, \mathbf{k}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-i\mathbf{x}(t) \cdot \mathbf{k}} e^{-i\omega t} d\mathbf{x} dt$$ \hspace{1cm} (4.7)

While this method does work to identify the occupied phonon modes in a system, a more effective metric can be used to provide greater contrast among the occupied modes by scaling the atomic density $\int_{-\infty}^{\infty} e^{-i\mathbf{x}(t) \cdot \mathbf{k}} d\mathbf{x}$ by the atomic velocity $\mathbf{v}$ in the direction $\mathbf{n}$ of interest as in Equation 3.35, which is repeated here for clarity as Equation 4.8.

$$I(\omega, \mathbf{k}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{v}(t) \cdot \mathbf{n} e^{-i\mathbf{x}(t) \cdot \mathbf{k}} e^{-i\omega t} d\mathbf{x} dt$$ \hspace{1cm} (4.8)

This method was recommended by Sebastian Volz and has proven to be more effective in revealing the phonon dispersion curves than the structure factor method; it is similar to computing the spectral energy density of the structure, as has been done by some researchers [115]. A side-by-side comparison of the results from each method can be found in Figure 4.13. Since all molecular dynamics simulations are actually discrete, the continuous form of the expression must be converted into a discrete form, as found in Equation 3.36, which is again repeated for clarity.
\[ I_k \{ \omega_j \} \approx \text{FFT}_j \left[ \sum_i v_i(t_j) \cdot n e^{-ix_i(t_j) \cdot k} \right] \] (4.9)

Using this discrete form, the phonon occupation of states can be computed along paths through the first Brillouin zone in different directions, resulting in the longitudinal and transverse phonon dispersion curves. The paths are selected so as to run along lines of superior symmetry, and the directions are selected to identify the longitudinal and transverse branches. Each branch (path and direction) must be computed separately; the multiple branches are combined by first normalizing the values in the branches for each wave vector \( k \), and then for each frequency taking the highest value among the branches. This way the most occupied modes for all materials of interest are shown in Figure 4.14, along with the available experimental data. It should be noted that while all occupied phonon modes should be identifiable within the first Brillouin zone, it was found while processing the simulation results that multiple equivalent paths needed to be considered to reconstruct the full set of dispersion curves; at least one path must be taken from a first Brillouin zone which is adjacent to the primary zone in reciprocal space. This is likely due to the limited size of the simulations possible with molecular dynamics.

As can be seen from Figure 4.14, when experimental data is available the simulations do
Fig. 4.14: Dispersion curves and density of states

(a) AlAs
(b) GaAs [116,117]
(c) InAs [118]
(d) AlGaAs
(e) AlInAs
(f) InGaAs
a very fine job of duplicating the optical and longitudinal acoustic branches of the dispersion
curves. The transverse acoustic branches are less successfully reproduced, with the present
work consistently over-predicting the resonant frequencies of the branches. This is again a
side effect of the imperfection of empirical potentials such as the Tersoff functional presently
used. In the case of InAs, the “experimental” results are actually \textit{ab initio} molecular
dynamics results using density functional theory (DFT). Attempts were made to correct
this discrepancy through modification of the potential coefficients, but these efforts were
not rewarded with any useful results.

Examination of Equation 4.9 reveals that considerable computation is required to com-
pute the dispersion curves from simulation results. This is largely because the spatial Fourier
transform responsible for conversion into reciprocal space cannot be expressed in a way that
allows for the application of the Fast Fourier Transform (FFT), which means the Discrete
Fourier Transform (DFT) must be computed directly; computation of the DFT requires
summation of values for each atom in the system for every time step for every wave vector
in the final dispersion plot. Since the number of wave vectors is proportional to the number
of atoms in the system, this computation scales as \( O(N^2) \), where \( N \) is the number of atoms
in the simulation. Making the situation even worse is the computational expense of the
exponential operator used in the Fourier transform. These scaling rates do not even include
the FFT component of the computation, which generally can scale as \( O(M \log M) \) where
in this case \( M \) is the number of time steps.

To handle this computational cost, the system output is first converted into the NetCFD
format for easier data access, and then each branch of the phonon dispersion is computed
separately, as defined by a set of wave vectors \( \mathbf{k}_i \) and a single normal vector \( \mathbf{n} \), as used in
Equation 4.9. Using OpenMP, the process branches into threads for each dispersion branch,
and loop through the time steps of the simulation together in lock-step. The master thread
reads a time step’s data into memory while the other threads wait for the I/O to complete.
After this is accomplished, they individually compute the input for a single wave vector and
single time, \( \sum_i \mathbf{v}_i(t_j) \cdot \mathbf{n} e^{-i\mathbf{k} \cdot \mathbf{x}(t_j)} \). This way, an input vector to the FFT is formed for each
wave vector simultaneously, and after all the inputs have been formed and stored in memory, the FFTs are executed in serial, which is ironically the fastest part of the computation. As described, this algorithm can utilize a number of cores less than or equal to the number of wave vectors in a dispersion relation, which has been adequate for the present purposes; the results presented here each took approximately four hours of execution time to compute on an AMD Phenom II X4 running at 3+ GHz, excluding the time to simulate the system itself. After the processing is complete, the results are stored in another NetCDF file which is considerably smaller, and can be quickly plotted using Python.

4.1.4 Thermal Conductivity Calculation

To compute the thermal conductivity of a material using molecular dynamics, there are two general approaches, the first being non-equilibrium methods which duplicate classical experiments and the second being equilibrium methods, which take advantage of relaxation relations to compute the conductivity from fluctuations within the system. Because of the strong size effects and unrealistic thermal gradients utilized in non-equilibrium methods, this work only uses the equilibrium calculation. This method is frequently called the “Green-Kubo” method, after the names of those who first developed the theory to make it possible [119–125]. The Green-Kubo relation, repeated here as Equation 4.10, allows for the computation of the thermal conductivity from the ensemble averaged autocorrelation of the equilibrium heat flux for a system.

\[
k = \frac{V}{k_B T^2} \int_0^\infty \langle \mathbf{J}(0) \cdot \mathbf{J}(t) \rangle \, dt \tag{4.10}
\]

The heat flux can be computed from atomic energies \(e_i\), velocities \(v_i\), and stress tensors \(\mathbf{S}_i\), as found in Equation 4.11. It should be noted that in many papers about this method, the units on \(\mathbf{J}\) are not consistent with those commonly used in engineering, and require special attention to make sure that everything is consistent; this generally involves the volume term \(V\) moving between the heat flux and the relaxation relation.
\[ J = \frac{1}{V} \left( \sum_i [e_i \cdot v_i] - \sum_i [\bar{S}_i \cdot v_i] \right) \] (4.11)

A theoretical derivation of the method is not be provided, but rather it is sufficient to say that the method has been applied successfully by numerous researchers to a variety of systems to compute the thermal conductivity of materials [34, 102, 119, 121]. While the terms in Equation 4.10 are straightforward to compute, being the system volume \( V \), temperature \( T \), and the Boltzmann constant \( k_B \), the ensemble averaged autocorrelation of heat flux requires special attention. Figures 4.17 through 4.22 show the flux autocorrelation functions as well as their integrals for the materials considered in this work. There are several ways that the integral in Equation 4.10 can be evaluated, but all of them require first computing the ensemble averaged autocorrelation of the system heat flux, as defined in Equation 4.12. A number of simulations of the same system with different initial velocity distributions are run and each system’s heat flux is autocorrelated according to equations 4.13 and 4.14.

\[
\langle f(t) \cdot f(0) \rangle = \frac{1}{R} \sum_{i=1}^{R} [A_i(\tau)] \] (4.12)

\[
A(\tau) = \int_{-\infty}^{\infty} f(t) f(t - \tau) \, dt \] (4.13)

\[
A\{j \Delta t\} = \frac{1}{M - j} \sum_{i+j}^{M} [f(t_i) f(t_i - j \cdot \Delta t)] \] (4.14)

After the ensemble averaged autocorrelation function has been computed from the set of simulations, the integral must be approximated somehow. An obvious choice is to numerically integrate the function, and take that as the value for use in the Green-Kubo calculation, but this introduces noise to the calculation that is undesirable; the values in the autocorrelation function become increasingly noisy with increasing \( \tau \) because fewer samples are averaged in the discrete computation as shown in Equation 4.14. This can easily be seen as the smooth numerical integral shown in the figures degenerates into increasingly erratic
noise for many of the materials. A method tried early on in the work was to simply identify
the plateau region of the numerically integrated function and use the average value to
compute the conductivity. While this works surprisingly well, it does not seem as rigorously
analytical as would be desired, and provides no time scale information about the relaxation
process.

The obvious and rigorous solution to these problems is to fit an analytical model to
either the autocorrelation function itself or its integral in order to compute the indefinite
integral and the time scales of the function. Initially, a single exponential function of the
form $A \exp[-t/\tau]$ was used to try and fit the autocorrelation function, but produced poor
results. This led to the idea of fitting the numerical integral of the autocorrelation function,
using the functional form given in Equation 4.15; the autocorrelation function from the fit
can be show using the function’s derivative, as in Equation 4.16.

$$f_I(t; A, \tau, p) = A (1 - \exp[-(t/\tau)^p]) \quad (4.15)$$

$$f_A(t; A, \tau, p) = \frac{df_I}{dt} = \frac{Ap^{p-1}}{\tau} \exp[-(t/\tau)^p] \quad (4.16)$$

This fitting form has proven to be remarkably robust and consistently able to return
a good fit using the *curve_fit* function from the scipy Python module; it is likely that the
robust fitting performance results from the orthogonality of the parameters in the fit, being
the asymptotic integral value $A$, the characteristic time $\tau$, and the order of approach $p$. The
only aspect of this function form which does not perform well is the derivative’s agreement
with the original autocorrelation function. Even though the fit matches the integral well,
it does not always match its derivative effectively, which leads to some questions about the
relevance of the characteristic time.

Dissatisfaction with this performance led to consideration of a second fitting functional
which is commonly found in the literature, but not initially used because of an obvious
failing in the form. Given by Equation 4.17, this form consists of a series of exponentials nearly identical to Equation 4.15, but all assuming an approach order of one. This results in multiple possible global minimums during the fitting procedure, as the coefficients can swap between the terms without changing the function at all. Normally only two terms are used, resulting in two peaks, and the derivative of the function is given by Equation 4.18.

\[
f_I(t; A_i, \tau_i) = \sum_{i=1}^{N} \left[ A_i (1 - \exp \left[ -t/\tau_i \right]) \right] \quad (4.17)
\]

\[
f_A(t; A_i, \tau_i) = \frac{df_I}{dt} = \sum_{i=1}^{N} \left[ \frac{A_i}{\tau_i} \exp \left[ -t/\tau_i \right] \right] \quad (4.18)
\]

In the case of both fit models, fitting both the autocorrelation function and its integral were attempted, and ultimately it was found that fitting the integral results in more stable results. Equation 4.15 is much easier to get good integral fits from, but with carefully selected initial guesses, Equation 4.17 can also fit the integral quite well. Upon a successful fit, however, Equation 4.18 always matches the autocorrelation function better than Equation 4.16, as seen especially well in the results for AlAs. It has been observed in the literature that fits of the second form are difficult, but an advantage of them is that they capture a fast and long time scale of phonon relaxation; there is some debate over the significance of these values, but in this work it was found that they always bracketed the characteristic time found by the other fitting method. The fitting parameters for the two functionals can be found in tables 4.3 and 4.4, while table 4.5 compares the results of the final fits with experimental values.

A discussion of the conductivities relative to experimental values are presented later, but it is encouraging to see that the thermal conductivities predicted by the two fitting functionals always fall very close to one another, and well within the error from the simulation itself. From this result, it is concluded that both fits produce equivalent results for thermal conductivity. It is interesting to note that the order of approach \( p \) is nearly the
Table 4.3: Fit 1

<table>
<thead>
<tr>
<th>Material</th>
<th>Mode Amplitude $A \left[ (eV/ps \cdot \AA^2)^2 \right]$</th>
<th>Decay Time $\tau \left[ ps \right]$</th>
<th>Decay Order $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlAs</td>
<td>$1.3 \times 10^{-5}$</td>
<td>77.8</td>
<td>0.86</td>
</tr>
<tr>
<td>GaAs</td>
<td>$4.6 \times 10^{-6}$</td>
<td>19.6</td>
<td>0.87</td>
</tr>
<tr>
<td>InAs</td>
<td>$2.7 \times 10^{-6}$</td>
<td>40.1</td>
<td>0.87</td>
</tr>
<tr>
<td>AlGaAs</td>
<td>$5.7 \times 10^{-7}$</td>
<td>2.39</td>
<td>0.85</td>
</tr>
<tr>
<td>AlInAs</td>
<td>$1.7 \times 10^{-7}$</td>
<td>2.27</td>
<td>0.36</td>
</tr>
<tr>
<td>GaInAs</td>
<td>$2.6 \times 10^{-7}$</td>
<td>2.99</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Table 4.4: Fit 2

<table>
<thead>
<tr>
<th>Material</th>
<th>Mode Amplitude 1 $A_1 \left[ (eV/ps \cdot \AA^2)^2 \right]$</th>
<th>Decay Time 1 $\tau_1 \left[ ps \right]$</th>
<th>Mode Amplitude 2 $A_2 \left[ (eV/ps \cdot \AA^2)^2 \right]$</th>
<th>Decay Time 2 $\tau_2 \left[ ps \right]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlAs</td>
<td>$1.22 \times 10^{-6}$</td>
<td>9.26</td>
<td>$1.15 \times 10^{-5}$</td>
<td>84.0</td>
</tr>
<tr>
<td>GaAs</td>
<td>$4.59 \times 10^{-7}$</td>
<td>2.31</td>
<td>$4.07 \times 10^{-6}$</td>
<td>21.6</td>
</tr>
<tr>
<td>InAs</td>
<td>$2.41 \times 10^{-7}$</td>
<td>4.58</td>
<td>$2.43 \times 10^{-6}$</td>
<td>43.3</td>
</tr>
<tr>
<td>AlGaAs</td>
<td>$9.58 \times 10^{-8}$</td>
<td>0.22</td>
<td>$4.88 \times 10^{-7}$</td>
<td>3.23</td>
</tr>
<tr>
<td>AlInAs</td>
<td>$5.72 \times 10^{-8}$</td>
<td>0.02</td>
<td>$1.05 \times 10^{-7}$</td>
<td>4.52</td>
</tr>
<tr>
<td>GaInAs</td>
<td>$9.29 \times 10^{-8}$</td>
<td>0.15</td>
<td>$1.60 \times 10^{-7}$</td>
<td>6.29</td>
</tr>
</tbody>
</table>

same for compounds as well as AlGaAs, but much lower for AlInAs and GaInAs. Also the characteristic times for the alloys are observed to be much lower than for the compound materials, as the uneven interactions caused by the alloying damp out the phonon oscillations more rapidly than in pure materials.

4.1.5 Potential Correction Explanation

When the potential coefficients for the Tersoff potential used by all the simulations in the present work were being found and encoded from the literature, there were initially problems with the optical branches of the phonon dispersion curves, and the thermal conductivity values predicted were much lower than expected, even for simulation results. The trouble was caused by a statement in the paper by Nordlund [51] that the parameter $\lambda_3$ was not used during that work, which was initially taken to mean that $\lambda_3 = 0$. This was not the case, but rather by “not used,” Nordlund must have meant that $\lambda_3 = 1$, which leaves the three-body aspect of the potential intact, whereas assuming zero changes the
The most puzzling aspect of this entire development is that the acoustic phonons are assumed to be responsible for the majority of the heat transfer in these systems since the group velocity of those phonons is much greater than that of the optical phonons. This can be seen from the dispersion curves and remembering that the group velocity for a traveling wave can be computed by Equation 4.19.

\[ \mathbf{v}_g = \frac{d\omega}{d\mathbf{k}} \]  

Thus, since the thermal conductivity can be expressed by \( k = \frac{1}{3} \rho C_v vl \), where \( \rho C_v \) is the volumetric heat capacity, \( v \) is the average energy carrier velocity, and \( l \) is the mean free path between collisions of that carrier, it is expected that those phonons with a higher group velocity contribute in more significantly than those with a low group velocity [126]. By inspection of the dispersion curves in Figure 4.14, it can be seen that the acoustic branches exhibit a much higher slope on average than the optical branches. The fact that modifying the potential in a way that corrected the optical branches would correct the thermal conductivity is puzzling, and lead to further investigation.

To measure which frequencies of phonon actually carry energy in the system, the spectral heat flux was computed by first computing the per-atom heat flux using the definition in Equation 4.11, but without the summation over atoms and division by volume, as shown in Equation 4.20.

\[ q_i = e_i v_i - \bar{S}_i \cdot v_i \]  

As can be seen from the equation, this is an expression for the total flux in a volume with the removal of the \( 1/V \) in the equation. The units on \( \bar{S}_i \) work out correctly because it is actually the atomic stress times the atomic volume. Normalizing the term by atomic volumes to achieve actual atomic stresses is difficult because of the complex definition for atomic volume in a deformed solid or liquid [95]. This term was used to replace the velocity.
in the density of states calculation to look at a density of heat flux, shown in Equation 4.21.

$$\text{SPQ} \{\omega\} \approx \sum_{k \in \text{atoms}} \left[ \sum_{i \in \{x,y,z\}} |\text{FFT}_{j} [q_k (t_j) \cdot n_i]|^2 \right]$$

(4.21)

After re-running all the dispersion curve calculations with the additional output of the per-atom heat flux the dispersion relations were re-plotted as Figure 4.15. There exists an interesting relationship between the spectral heat flux and the density of states, as well as the cloudy regions in the dispersion relations that were previously assumed to be caused by noise within the simulation. As in the previous plots, the red line in the DOS region plots the density of states, but the dashed red line is the same curve plotted at double the standard frequency to look for harmonics of the first-order vibration. The spectral heat flux is shown as the blue solid line, and can be seen to strongly correlate with the density of states first harmonic in for the acoustic branches, but the second harmonic for the optical branches.

This mixed correlation is not surprising after the differing nature of acoustic and optical phonons is considered. While acoustic phonons are literally sound waves, and it can be demonstrated that the low wave vector acoustic branch group velocities correspond to the macro-scale acoustic speeds for solids, optical phonons are the result of back and forth motion by atoms of opposing electro-negativity. This characteristically different motion must push the frequencies of heat transfer to the second harmonic of oscillation, which explains the correlations between the spectral heat flux and the density of states at different harmonics. Figure 4.16 shows the relationship a little more clearly by plotting only the density of states (both harmonics) and the spectral heat flux.

While these results and reasoning do not explain every aspect of the plots, it is sufficient to explain why the optical branches seem to be so important to the heat transfer in these systems. It can be seen that much of the heat transfer occurs in the second harmonic of the optical branches, so by correcting the potential in $\lambda_3$, all of the higher frequencies were corrected and not just the optical branches as was observed. After further consideration of
(a) AlAs

(b) GaAs

(c) InAs

(d) AlGaAs

(e) AlInAs

(f) InGaAs

Fig. 4.15: Dispersion curves and density of states with spectral heat flux
Fig. 4.16: Density of states and spectral heat flux
materials in the literature and with these results in mind, a value of $\lambda_3 \cong 1.5$ was selected for the potential.

4.1.6 Thermal Conductivity Results for Materials of Interest

Figures 4.17 through 4.22 show the simulation results from all the materials which can be simulated using the current potential coefficient set. From the figures it can again be seen that the two-term fitting form does a better job of matching the autocorrelation function, but that both fitting forms perform well for fitting the integral itself; the thermal conductivity is computed only from the asymptotic value of the integral, so its calculation in not strongly affected by the form of the fit in these cases.

The results from the figures are summarized in table 4.5, which also compares the results with experimental values for the bulk properties of the materials. It can be seen from the table that the predicted thermal conductivities from each fitting method agree quite well with each other, but that there are moderate disagreements with the experimental values for each material.

Generally, the simulation results under-predict the thermal conductivity of the samples, which is a commonly-encountered aspect of molecular dynamics simulations characteristic called size effects. Some of the simulations were re-run on a series of system sizes to explore this possibility, only to find that while the systems resulted in different conductivity predictions, they were not consistently higher or lower than the values reported in the table. The results from the size effects investigation are reported in table 4.6.

<table>
<thead>
<tr>
<th>III-V Material</th>
<th>Experimental $k$ [$W/m \cdot K$]</th>
<th>Fit 1 $k$ [$W/m \cdot K$]</th>
<th>Error [%]</th>
<th>Fit 2 $k$ [$W/m \cdot K$]</th>
<th>Error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlAs</td>
<td>[127]80</td>
<td>105.8</td>
<td>+32</td>
<td>103.8</td>
<td>+30</td>
</tr>
<tr>
<td>GaAs</td>
<td>[128]55</td>
<td>37.1</td>
<td>-33</td>
<td>36.6</td>
<td>-33</td>
</tr>
<tr>
<td>InAs</td>
<td>[128]27</td>
<td>27.0</td>
<td>0</td>
<td>26.5</td>
<td>-2</td>
</tr>
<tr>
<td>AlGaAs</td>
<td>[129]9.0</td>
<td>4.65</td>
<td>-48</td>
<td>4.75</td>
<td>-47</td>
</tr>
<tr>
<td>AlInAs</td>
<td>[105,130]3.5</td>
<td>1.56</td>
<td>-55</td>
<td>1.48</td>
<td>-58</td>
</tr>
<tr>
<td>GaInAs</td>
<td>[131,132]5</td>
<td>2.32</td>
<td>-54</td>
<td>2.29</td>
<td>-54</td>
</tr>
</tbody>
</table>
Fig. 4.17: Thermal conductivity of AlAs

Fig. 4.18: Thermal conductivity of GaAs

Fig. 4.19: Thermal conductivity of InAs
Fig. 4.20: Thermal conductivity of AlGaAs

Fig. 4.21: Thermal conductivity of AlInAs

Fig. 4.22: Thermal conductivity of GaInAs
Table 4.6: Study of size effects for InAs

<table>
<thead>
<tr>
<th>Side Length</th>
<th>Fit 1</th>
<th>Fit 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Cells</td>
<td>$k [W/m \cdot K]$</td>
<td>$k [W/m \cdot K]$</td>
</tr>
<tr>
<td>6</td>
<td>26.95</td>
<td>26.51</td>
</tr>
<tr>
<td>7</td>
<td>35.95</td>
<td>37.26</td>
</tr>
<tr>
<td>8</td>
<td>32.52</td>
<td>33.48</td>
</tr>
<tr>
<td>10</td>
<td>27.77</td>
<td>26.78</td>
</tr>
<tr>
<td>12</td>
<td>25.29</td>
<td>25.07</td>
</tr>
</tbody>
</table>

As can be seen from the results in the table, there is variation in predicted thermal conductivity with changes to the system’s size, but these variations are not uniform; they are more likely a result of the imperfect ensemble averaging as represented in the uncertainty region of the autocorrelation integral plots. While similar studies have not been conducted for the other materials considered, it is assumed that if there are size effects present in the simulations, that their effects are relatively minimal. One of the advantages of using the Green-Kubo method with equilibrium molecular dynamics is a weakening of size effects relative to those seen with non-equilibrium simulations. The present results seem to support this general trend observed in the literature.

4.2 Computation Results for a QCL Period

Although a firm rejection or acceptance of the potential is not made, the validation efforts suggest that it can be used with the understanding that thermal properties it predicts may vary significantly from the true values; in the case of thermal conductivity (which is the primary property of interest) this variation is likely to be on the order of 50%. Since improvement of the potential coefficients with regard to the thermal conductivity is beyond the scope of this work, the QCL was simulated using the potential as-is, and understanding its limitations.

The QCL system was run through many of the same simulations as the bulk materials, and the results of these simulations are discussed below.
4.2.1 Thermal Expansion and Specific Heat

The lattice parameter and thermal expansion results for the QCL simulations are shown in Figure 4.23. Comparison of these results with those of the bulk materials shows that the QCL system most resembles that of GaAs and GaInAs in this respect. Based on the GaAs simulation and experimental data, it is likely that the QCL thermal expansion is over-predicted by a factor of 2-3.

Figure 4.24 shows the specific heat simulation results for the QCL, which is predicted to have a specific heat of 327 $W/kg \cdot K$ at 300 $K$. Again, this result follows GaInAs fairly closely, and from the validation results for the prediction of specific heats of materials, this result should be off by only 10-15 % at the worst.

4.2.2 Dispersion and Density of States

The results of the density of states and dispersion calculations for the QCL system are shown in Figure 4.25. It can be seen in Figure 4.25a that the spectral heat flux follows a similar pattern in the QCL as in many of the bulk materials, most especially AlInAs. Perhaps more interesting though, is the formation of multiple optical bands in the dispersion curves shown in Figures 4.25b through 4.25d; this is caused by the presence of both GaInAs and AlInAs in the structure, which resonate at very different frequencies for the optical mode. This difference of optical mode frequency can be easily explained by the different masses of Al and Ga. There are two blocks of dispersion curves in each plot; those on the left are calculated in the in-plane directions, while those on the right are calculated in the cross-plane directions. Examination of the differences between these two groups can yield insight into the effects of the interfaces in the cross-plane direction. It should also be noted that the in-plane directions are much smaller in the simulation, and therefore are composed of fewer points in the plots because the system supports fewer oscillatory modes in those directions.

Also visible is the loss of clear harmonics in the dispersion curve as also seen in AlInAs and GaInAs. The ghosted bands in the dispersion relations can be seen to correspond with harmonics of the optical modes just as with the bulk material simulations. The dispersion
(a) Lattice parameter versus temperature  
(b) Thermal expansion coefficient versus temperature

Fig. 4.23: Thermal expansion coefficient from QCL simulations

(a) Energy versus temperature  
(b) Specific heat versus temperature

Fig. 4.24: Specific heat from QCL simulations
branches are plotted separately here for clarity because of the loss of clear harmonics in the acoustic branches, which was also observed in the simulations of AlInAs and GaInAs.

4.2.3 Thermal Conductivity

Using the same techniques as described for the bulk materials computations used to validate the potential, the thermal conductivity of the QCL has been computed along the cross-plane and in-plane directions. While running the simulations it was discovered that the relaxation times for the cross-plane direction were long enough to require a longer simulation time than that used for the bulk simulations, but other than the increased simulation time and consideration of separate directions, the simulation procedures were identical.

The results for the in-plane conductivity are shown in Figure 4.26, where it can be seen that the results bear a striking resemblance to the results for both AlGaAs and AlInAs. This result is not terribly surprising, since the QCL is partially composed of AlInAs, and near the interfaces between layers the structure is very similar to AlGaAs.

The cross-plane conductivity results are shown in Figure 4.27, and while they seem reasonable, they are more difficult to interpret. An obvious feature is the disagreement between the conductivity predictions made by the two fitting methods, which normally agree very well; even though the first fitting form seems to match the autocorrelation integral more perfectly, the second fitting method matches the autocorrelation itself much better. The difference between the two fitting methods is still small relative to the uncertainty present in the ensemble averaging itself, and the actual uncertainty due to the imperfections of the inter-atomic potential make the difference between the fitting methods inconsequential.

It is difficult to understand how a large system composed of multiple layers with imperfect interfaces can demonstrate a thermal conductivity greater than that of its constituent layers. Rather, it is expected for the conductivity of the layered system to be reduced by the interfaces as they scatter phonons propagating between the layers. It is possible that the increase in predicted conductivity is a result of size effects that were not observed by the previously-conducted study, since none of those systems were quite so large as the QCL (which is 78 unit cells in length).
Fig. 4.25: Dispersion and density of states data for QCL simulations

(a) Density of states and spectral heat flux

(b) Longitudinal dispersion relations

(c) Transverse mode 1 dispersion relations

(d) Transverse mode 2 dispersion relations

Fig. 4.26: QCL in-plane thermal conductivity

(a) Autocorrelation

(b) Autocorrelation integral
Because of this discrepancy, additional simulations were run on the AlInAs and GaInAs systems, but instead of using small cubic geometries as previously run for the potential validation, the systems were constructed to match the QCL system in each dimension. The conductivities of these two systems were then computed along the same long axis as the cross-plane direction in the QCL systems. The results of these two simulation sets are found in Figures 4.28 and 4.29.

As expected, the larger systems demonstrate an increased thermal conductivity, which can be easily explained as the result of size effects. The larger system can support much longer phonon wavelengths than the small system, resulting in a richer phonon population capable of energy transport. These results also shed light on the QCL cross-plane results, which previously seemed to defy logic by showing a conductivity higher than the compounds from which the QCL is composed. The QCL results are lower than either the AlInAs or GaInAs results, which supports the theoretical prediction that the interfaces should scatter phonons as they are encountered and thereby reduce the overall cross-plane conductivity. These results, as well as a projection of the expected physical thermal conductivity for the QCL in the in-plane and cross-plane directions are found in table 4.7.

The QCL projected thermal conductivity ranges were computed by scaling the simulation results by the ratio of the experimental value to the simulation result for the compounds
Fig. 4.28: AlInAs QCL-geometry thermal conductivity

Fig. 4.29: GaInAs QCL-geometry thermal conductivity

Table 4.7: Summary of QCL simulation results

<table>
<thead>
<tr>
<th>Material</th>
<th>Geometry</th>
<th>Experimental $k$ [W/m·K]</th>
<th>Simulation $k$ [W/m·K]</th>
<th>Ratio</th>
<th>Projection $k$ [W/m·K]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlInAs</td>
<td>Cube</td>
<td>3.5</td>
<td>1.48</td>
<td>2.36</td>
<td>3.71-4.01</td>
</tr>
<tr>
<td>GaInAs</td>
<td>Cube</td>
<td>5</td>
<td>2.29</td>
<td>2.18</td>
<td></td>
</tr>
<tr>
<td>QCL</td>
<td>In-Plane</td>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AlInAs</td>
<td>QCL</td>
<td>3.5</td>
<td>10.3</td>
<td>0.340</td>
<td>1.84-4.29</td>
</tr>
<tr>
<td>GaInAs</td>
<td>QCL</td>
<td>5</td>
<td>6.3</td>
<td>0.794</td>
<td></td>
</tr>
<tr>
<td>QCL</td>
<td>Cross-Plane</td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
in each direction. The lower bound and mean value on the cross-plane direction is much smaller than the lower bound on the in-plane direction, which supports the boundary scattering theory. Furthermore, the projected in-plane conductivity range lies between the experimental values for AlInAs and GaInAs, which is to be expected without the presence of interfaces to scatter the phonons as they travel. The lower bound and mean value of the cross-plane QCL projection are also both lower than the experimental conductivities for the composite materials, which also supports the interface scattering theory.

As noted in the introduction, there have been many measurements of the effective thermal conductivity of QCLs’ active regions, which have all resulted in values in a range from $1.8 \, W/m \cdot K$ to $8 \, W/m \cdot K$ [13, 15–20]. The experimental and simulation conductivity results are compared in Figure 4.30. While none of the QCLs measured in these efforts exactly match the system simulated in this work, most were composed of AlInAs/GaInAs, and possessed similar layer structures and even layer densities. This being the case, it can be stated that molecular dynamics simulations seem capable to compute the effective bulk thermal conductivity of a QCL’s active region from only the layer structure information and inter-atomic potential coefficients to within the experimental and simulation errors of each method. While the potential can and should be improved, and there are still questions concerning the possibility of size effects in the QCL conductivity calculation, it is certain that the method is valid and deserves further investment in terms of time and resources from the QCL community.

4.3 Photothermal Uncertainty Analysis

To evaluate the expected uncertainty in measurements of thermal properties made using photothermal radiometry and similar methods, a Monte Carlo uncertainty analysis was performed using the PTR system at Utah State as a model for operation. The statistical distribution of various system parameters was measured in the lab, and then the biases removed as calibration, resulting in Gaussian distributions for each input to the PTR data reduction process which could be modeled as a single uncertainty based on the standard deviation; the results of this system evaluation as well as other assumed uncertainties are
The measurement mode to be tested is the most commonly encountered use of PTR, that is, measurement of the thermal properties of an unknown film on a known substrate. A variety of films and substrates were simulated to ensure that the results are more general than a single sample. Reported here are the trends and average results of those set of samples.

Using these input distributions, a population of input systems was simulated to produce a set of frequency scans which represent the set of all scans expected to potentially come from a system in this population. Each frequency scan is then fit assuming that there is no uncertainty in the measurements. The differences between the parent input distribution and

Table 4.8: Uncertainty inputs to Monte Carlo simulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Thickness</td>
<td>2000.0</td>
<td>µ</td>
</tr>
<tr>
<td>Film Thickness</td>
<td>10.0</td>
<td>µ</td>
</tr>
<tr>
<td>Beam Diameter</td>
<td>1000.0</td>
<td>µ</td>
</tr>
<tr>
<td>Detector Spot Diameter</td>
<td>200.0</td>
<td>µ</td>
</tr>
<tr>
<td>Frequency Uncertainty</td>
<td>0.0083</td>
<td>Hz</td>
</tr>
<tr>
<td>Film Thickness Uncertainty</td>
<td>5.0</td>
<td>%</td>
</tr>
<tr>
<td>Phase Uncertainty (Lock-in)</td>
<td>1.0</td>
<td>°</td>
</tr>
<tr>
<td>Phase Uncertainty (Transfer Function)</td>
<td>0.5</td>
<td>°</td>
</tr>
<tr>
<td>Diffusivity Uncertainty</td>
<td>4.75</td>
<td>%</td>
</tr>
<tr>
<td>Effusivity Uncertainty</td>
<td>3.0</td>
<td>%</td>
</tr>
</tbody>
</table>
the final child population of frequency scan fits reveals information about the uncertainty of the final result due to the input uncertainties and the data reduction process.

### 4.3.1 Monte Carlo Results

One of the first discoveries from the data was that the uncertainty in the diffusivity due to the uncertainty in the film thickness is governed by a non-linear relationship, as demonstrated by the difference in shape between the diffusivity distributions in the output population when compared with a normal distribution as seen in Figure 4.31b. Upon further investigation, this can be clearly seen from sensitivity analysis done on a single theoretical sample, as shown in Figure 4.31a.

It can be seen from the plot that while there is no error in the computed effusivity in this case, the diffusivity error increases with the variation in film thickness from the value assumed during the fit. Also, the non-linearity in the relationship can be clearly observed in the curvature of the diffusivity error line.

One goal of the uncertainty analysis was to evaluate two frequency sampling methods for the frequency scan. From the form of the solutions to the thermal wave field equation, it is suggested that the correct sampling mode for the frequencies in a scan is a power law or logarithmic sampling. While it may be enough to believe this assessment, with the present method it was not difficult to setup a test to evaluate the relative merits of logarithmic spacing, as suggested by the solutions to the thermal wave field equation, and linear spacing, which is the assumed default for any sampling. This was done by computing the film diffusivity and effusivity uncertainties as a function of the number of samples in the frequency for both logarithmic and linear spacing. The results of this study are shown in Figure 4.32.

It can be seen from the figure that the nominal bias for both properties is very low at high sample counts, independent of the sampling method. This cannot be said for low sample counts, where the logarithmic spacing demonstrates the same uncertainty behavior as for high sample counts, but the linear sampling uncertainty dramatically increases. A similar trend can be seen in the precision data for both properties, although the transition
Fig. 4.31: Demonstration of nonlinear uncertainty effects in diffusivity

Fig. 4.32: Bias and precision uncertainty due to number of samples
is more gradual in this case. From these results it can be concluded that the original assessment of spacing requirements is correct: logarithmic spacing is the correct sampling mode for frequency in the frequency scanning process.

The final investigation conducted as part of the uncertainty evaluation is an assessment of how variation in the film thickness uncertainty affects the uncertainty of the computed thermal properties. The films measured by PTR can be very thin, extending into the micro-scale. For this reason, the film thickness may not be very well known, and the affects of its magnitude should be understood. The results of this study are shown in Figure 4.33.

The bias uncertainty for both thermal properties is quite small, even for large uncertainties in the film thickness. This trend is not, however, echoed in the results for the precision uncertainty. The uncertainty of the film diffusivity increases rapidly and linearly as the film thickness uncertainty increases; with a modest 6% uncertainty in film thickness, the expected uncertainty in the diffusivity is almost twice that. The results for the effusivity uncertainty are completely dissimilar, with the effusivity precision uncertainty being entirely unaffected by the film thickness uncertainty.

### 4.4 PTR Results from QCL Frequency Scans

Using photothermal radiometry, the phase versus frequency response of two quantum cascade laser diodes have been measured, as shown in Figure 4.34. These samples were provided by the THALES corporation of France to hopefully provide validation to the molecular dynamics simulations being conducted.

The two samples are identified as *large* and *small*, based solely on the diameter of the samples before they were split for scanning. Both samples were scanned using a small diameter heating laser, as well as a laser which was first passed through a diffuser to dramatically widen its effective area. The larger heating area is required for low frequencies to prevent the occurrence of three dimensional heat transfer in the sample. Using the flat laser in the low frequencies and the point laser in the higher frequencies allows the one dimensional heat transfer model to be used at all frequencies in the scan, and prevents the need for measuring beam diameters and detector spot sizes, or the absolute requirement
Fig. 4.33: Bias and precision uncertainty due to film thickness

(a) Bias

(b) Precision

Fig. 4.34: Phase response data for QCL samples

(a) Front face

(b) Back face
of concentric alignment of the laser and detection spot. Both samples were also scanned on the front and back sides of the sample. This should provide contrast and aid in the interpretation of the results. A simple analysis of the front-face frequency scans is shown in Figure 4.35, but provides only qualitative conclusions.

The limit between the thermally-thin and thermally-thick regimes is designated, and can be clearly distinguished by the coalescence of the flat and point heating modes’ phase response data. In the low frequency range, the flat data is the result of one-dimensional heat conduction, and so should be used for analysis when fitting against one-dimensional models, while the point heating results in two-dimensional axisymmetric temperature profiles which cannot be adequately modeled without considerable extra data that is not available. At higher frequencies, the signal caused by the thin film on the substrate is designated as such.

4.4.1 Curve Fitting Attempts

A variety of curve fitting algorithms and processes were used in pursuit of good fits to the experimental data, but all were met with only qualified success. The best results from these efforts are shown in Figure 4.36.

After careful consideration of the heat transfer during the PTR frequency scanning process, and by comparing theoretical models with the experimental data, it was concluded that the phase data for frequencies below 100 Hz are not correctly modeled with the quadrupole methods used in this work, and so to prevent spurious modes from entering the fitting process, phase data for these frequencies was discarded as can be seen by comparing Figure 4.36 with Figure 4.34.

While the initial fitting attempts made use of a novel residual definition based on the sensitivity of the phase to each fit parameter, these methods were abandoned in favor of a traditional least-squares approach. It is believed that while the new method demonstrates promise, it would require the development of specialized optimization algorithms that could take advantage of the additional information provided by the specialized fitting residuals; these algorithms are outside the scope of the present work and should be returned to for later investigation.
Fig. 4.35: Cursory analysis of QCL frequency scans

(a) Large sample

(b) Small sample

Fig. 4.36: Phase response data for QCL samples with fitting attempts
Despite the fact that they are not needed for the fitting procedure, the sensitivities of the phase to each fitting parameter are still included in the plots to provide insight into the fit and the effect of each parameter. The sensitivity in this context is defined as the derivative of the phase with regard to the parameter, normalized by the parameter’s value, as in Equation 4.22.

\[ S_p = \left. \frac{\partial (\angle \theta)}{\partial p} \right|_p \cdot p \tag{4.22} \]

In this equation, \( p \) represents any of the fitting parameters, while \( \angle \theta \) is the sample surface phase as measured by experiment or predicted by the model. By inspection of the expression, it can be seen to have units of angle, and so is plotted alongside the phase results and lines of best fit.

To best fit the experimental data, an appropriate parameter space should be used during the fitting process. In many cases, there exist non-dimensional numbers which provide such a space, and thus the present problem is analyzed in search of its non-dimensional numbers. This is accomplished by expanding a simple one-dimensional two-layer quadrupole solution from its impedance matrix form into the final expression for the front surface phaser (complex representation of amplitude and phase) temperature as a function of the laser heat flux on that same surface, as shown in equations 4.23 through 4.26.

\[
\theta_f = H_0 \frac{b \cosh \left( \frac{1+i}{\sqrt{2} F_0 F} \right) \cosh \left( \frac{1+i}{\sqrt{2} F_0 S} \right) + \sinh \left( \frac{1+i}{\sqrt{2} F_0 F} \right) \sinh \left( \frac{1+i}{\sqrt{2} F_0 S} \right)}{b \sinh \left( \frac{1+i}{\sqrt{2} F_0 F} \right) \cosh \left( \frac{1+i}{\sqrt{2} F_0 S} \right) + \cosh \left( \frac{1+i}{\sqrt{2} F_0 F} \right) \sinh \left( \frac{1+i}{\sqrt{2} F_0 S} \right)}
\]

\[ H_0 = \phi_f \exp \left( -i \frac{\pi}{4} \right) \frac{1}{\epsilon F \sqrt{\omega}} \tag{4.24} \]

\[ b = \frac{\epsilon F}{\epsilon S} \tag{4.25} \]

\[ F_{0i} = \frac{\alpha_i}{\omega L_i^2} \tag{4.26} \]

In these equations, \( \theta_f \) is the front face phaser temperature, \( \phi_f \) is the heat flux from the laser on the front face, \( i \) is the layer index (\( F \) for film and \( S \) for substrate), \( \epsilon_i \) is the layer
thermal effusivity, $\alpha_i$ is the layer thermal diffusivity for layer $i$, $L_i$ is the layer thickness, and $\omega$ is the laser modulation frequency. The leading term, $H_0$, shows the temperature amplitude to be directly proportional to laser flux $\phi_f$, and inversely proportional to the square root of the laser modulation $\omega$. It is for this reason that PTR amplitude data is often reported after multiplication by $\sqrt{\omega}$, since the contribution is universal and independent thermal properties. The laser flux, $\phi_f$, is a function of laser power, beam geometry, and surface properties; the complex heat transfer involved precludes the quantitative usefulness of amplitude data from PTR measurements for most applications. The leading term also depends on the film effusivity, but since the laser flux is generally unknown, this dependence is of little practical value. Most interestingly, the leading term contains the phase of $-45^\circ$ in the $\exp(-i\pi/4)$ term, which is the asymptotic phase for a thermally-thick material.

All the interesting phase behavior is a result of the ratio term in Equation 4.23, which can be seen to depend on only standard functions of three expressions. The first and simplest, is the ratio of the layers effusivities, $b$, whose interpretation is straightforward. The second two, $F_{oF}$ and $F_{oS}$, are the Fourier functionals for the film and substrate respectively. While normally heat transfer is non-dimensionalized in time using the Fourier number, in this case it is the frequency rather than the time that is used for non-dimensionalization, as shown in Equation 4.26. Since multiple frequencies are considered simultaneously, it is more appropriately termed a functional in this case, but regardless of its appellation, the relevant physical parameters in the functional are the thermal diffusivity and the layer thickness. Since these two properties never occur in any other combination within the solution, there is no way to distinguish variations in one from the other in experimental data.

From this analysis, it is concluded that the correct fitting space for PTR data is in the form of the effusivity ratios and the Fourier functional for each layer. To these values an additional thermal resistive layer of value $R_i$ is assumed to exist between the layers which must also be fit. The fits were conducted with results very similar to those presented earlier; such unimpressive fitting results encouraged further investigation, and so the model was expanded to include the effects of semitransparent absorption of the laser instead of
restricting its influence to the surface alone. From the work of Salazar et al, a quadrupole model for such a scenario can be readily found [71,72]. Applying the model to the two-layer with resistance assumption made here results in equations 4.27 though 4.33.

\[ \vec{\theta}_f = \bar{Z}_F \left( \bar{R} \bar{Z}_S \left( \vec{\theta}_b - \bar{X}_S \right) - \bar{X}_F \right) \]  
\[ \vec{\theta}_j = \begin{cases} \theta_j \\ \phi_j \end{cases} \]  
\[ \bar{Z}_i = \begin{bmatrix} \cosh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) & (\epsilon_i \sqrt{i\omega})^{-1} \sinh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) \\ (\epsilon_i \sqrt{i\omega})^{+1} \sinh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) & \cosh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) \end{bmatrix} \]  
\[ \bar{R} = \begin{bmatrix} 1 & R_i \\ 0 & 1 \end{bmatrix} \]  
\[ \bar{X}_i = \begin{cases} C_i \left( \frac{A_i}{\sqrt{\omega \alpha_i}} \sinh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) - \cosh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) + \exp (-A_i L_i) \right) \\ C_i k_i \left( \frac{\omega \alpha_i}{i \omega \alpha_i} - A_i \cosh \left( \frac{1\!+\!i}{\sqrt{2} \alpha_i} \right) + A_i \exp (-A_i L_i) \right) \end{cases} \]  
\[ C_i = \frac{I_i A_i}{2k_i (\omega \alpha_i - A_i^2)} \]  
\[ I_i = I_1 \exp \left( -\sum_{j=1}^{i-1} A_j L_j \right) \]

In these equations all terms are as previously defined, with the addition of \( A_i \) as the layer absorptivity and \( I_i \) as the incident light on each layer. After evaluation and simplification of the equations, the final form can be expressed as in Equation 4.34, which leads to the final solution in Equation 4.35.

\[ \begin{cases} \theta_F \\ 0 \end{cases} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{cases} \theta_B \\ 0 \end{cases} - \begin{cases} x \\ y \end{cases} \]  
\[ \theta_F = \frac{a}{c} y - x \]
In Equation 4.34, the impedance matrix and the subtracted vector are the results of matrix operations on Equation 4.27. This expression can be easily converted into a computer algorithm to calculate the thermal response of arbitrary layered systems with volumetric absorption. After implementation in Fortran2003, it was found that the expression results in excessive round-off floating-point errors which make the results useless. This was not unexpected, since the non-absorptive model suffers from similar problems, but the inclusion of the absorption vector in the solution makes the methods normally used to stabilize the numerics in the equations no longer applicable. Significant efforts to stabilize the expressions to enable their evaluation using hardware floating point methods were expended before deciding to solve the problem using a multi-precision library in Python. After translation, the naive version of the algorithm worked perfectly, but requires the inclusion of no less than 100 significant decimal digits to provide sufficient accuracy for stable results. Based on this, it is unsurprising that the 33 or so significant digits provided by quadruple precision in Fortran were insufficient to express the results. The results of the fit are summarized in table 4.9, including conversion of the fitted parameters into more commonly-encountered thermal properties.

While neither curve fit is of sufficient quality to be truly trusted, the fit results for the large sample may at least be compared with the simulation results presented earlier for QCL systems. Using the simulation results, and a computed density of 5286 kg/m$^3$, the QCL mean predicted thermal diffusivity is $1.77 \times 10^{-6} \text{ m}^2/\text{s}$, which is almost four times the thermal diffusivity from the curve fit. Given that the simulation system is not identical to the experimental material, and the poor quality of the fit, very little can be said for the result other than that it is at least on a similar order of magnitude. It has been suggested by some that the unusual phase response of the large sample may be the result of plasmon excitation on the sample surface, but without considerable additional effort, there is no way to truly know the cause.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Units</th>
<th>Large Sample</th>
<th>Small Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega \cdot F_o_f$</td>
<td>$s^{-1}$</td>
<td>$5.267 \times 10^5$</td>
<td>$1.000 \times 10^7$</td>
</tr>
<tr>
<td>$b$</td>
<td>-</td>
<td>4.004</td>
<td>0.276</td>
</tr>
<tr>
<td>$r</td>
<td>R_i = 10^r$, $a</td>
<td>A_f = 10^{-a}$</td>
<td>$m^2 \cdot K/W$, $m^{-1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.05</td>
<td>5.802</td>
</tr>
<tr>
<td>$\omega \cdot F_o_s$</td>
<td>$s^{-1}$</td>
<td>$3.020 \times 10^2$</td>
<td>$3.020 \times 10^2$</td>
</tr>
<tr>
<td>$L_f$</td>
<td>$\mu m$</td>
<td>0.945</td>
<td>6.24</td>
</tr>
<tr>
<td>$L_s$</td>
<td>$\mu m$</td>
<td>350</td>
<td>350</td>
</tr>
<tr>
<td>$\alpha_f$</td>
<td>$m^2/s$</td>
<td>$4.703 \times 10^{-7}$</td>
<td>$3.894 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\alpha_s$</td>
<td>$m^2/s$</td>
<td>$3.700 \times 10^{-5}$</td>
<td>$3.700 \times 10^{-5}$</td>
</tr>
<tr>
<td>$R_i$</td>
<td>$m^2 \cdot K/W$</td>
<td>$5.309 \times 10^{-8}$</td>
<td>$8.610 \times 10^{-8}$</td>
</tr>
<tr>
<td>$A_f$</td>
<td>$m^{-1}$</td>
<td>$8.913 \times 10^{-11}$</td>
<td>$1.578 \times 10^{-6}$</td>
</tr>
<tr>
<td>$A_s [133]$</td>
<td>$m^{-1}$</td>
<td>$1.0 \times 10^{-6}$</td>
<td>$1.0 \times 10^{-6}$</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusions

While there will always be opportunities to push analysis tools to new levels of accuracy and capability, they can still provide useful information without perfection. That is certainly the case with this work; the inter-atomic potential does very well with some properties such as specific heat and dispersion relations, adequately with others such as thermal conductivity, and less well with the thermal expansion coefficient. The results are consistent enough that fairly confident predictions can be made for the thermal properties of the QCL structure. Additional effort could be spent on improving the potential coefficients for thermal properties, and the size effects in the thermal conductivity could be explored, but as the work stands there are already useful results that can help improve the development of QCL diodes.

Similarly, the photothermal radiometry experiments can and will be improved, especially with additional scans and fits; more sophisticated models should be used to explore the reasons that the standard PTR fitting models could not match the experimental results. However, the frequency scans taken of the samples are the first use of PTR on such structures, and this effort marks the beginning of more direct measurements of QCLs’ thermal properties than has been previously accomplished.

To conclude, the previously-stated objectives will be reviewed; it is shown that all objectives have been addressed in the results section.

1. Use molecular dynamics to compute the effective thermal conductivity of a characteristic quantum cascade laser diode super-lattice structure.

   - Evaluate inter-atomic potential by simulation and conductivity calculation of III-V bulk materials in QCL composition and alloys thereof.
The thermal properties of all the materials related to the QCL structure have been computed and compared with experimental results when available. Size effects in the thermal conductivity calculations have been identified and explained.

- Compute the cross-plane QCL thermal conductivity.

The cross-plane QCL thermal conductivity has been calculated, and using the size effect information for AlInAs and GaInAs, an expected value has been computed; this value lies within the range of QCL cross-plane conductivities from experiments as found in the literature, suggesting that the MD approach to predicting the QCL conductivity shows merit. Further development of interatomic potentials and quantification of size effects could be used to enhance the precision of the predictions.

- Compute the in-plane QCL thermal conductivity.

Similar to the cross-plane results, the in-plane QCL conductivity has been computed and compared with the experimental conductivity of AlInAs and GaInAs, since no in-plane QCL experimental data is available. The projected value for the in-plane QCL conductivity lies between the experimental values for AlInAs and GaInAs, with is expected from theoretical considerations.

- Compare results with literature.

Literature comparisons have been made as noted previously.

2. Construct a photothermal radiometry system.

- Verify correct operation of system.

The PTR system constructed at Ecole Centrale de Paris has been vetted by Dr Mihai Chirtoc, a long-time expert in the field of photothermal experimentation, and found to function normally. Work to calibrate and utilize the system is ongoing by graduate students in Paris.

- Assess the uncertainty trends in PTR systems using Monte Carlo uncertainty analysis.
The general uncertainty performance of PTR based on the characteristics of a system at Utah State University has been evaluated using the Monte Carlo uncertainty method. The results are reported previously in this work. The uncertainty results are found to follow the expected trends from careful examination of solutions to the one-dimensional thermal wave field equation, suggesting that simpler relations can be used to analyze the uncertainties of measurements made in the future with different systems.

- Run frequency scans on QCL; interpret results.

Because of frequency limitations of the PTR system at Ecole Centrale, the QCL samples provided by Thales were scanned using the PTR system in Reims France in the lab of Dr Mihai Chirtoc and with his assistance. While the system at Ecole Centrale de Paris can only scan up to 250 kHz, the system at Reims can scan to 1 MHz, which is needed for samples with such thin films and seen in the previous results. The resulting frequency scans have been interpreted and fitting attempts made with a variety of thermal transport models. Additional work will be needed to fully understand the results.
References


Appendices
Appendix A
Supporting Data and Derivations

A.1 Measured Radiation for 2D Axisymmetric PTR

A.2 Thermal Expansion

\[ \alpha_L = \frac{1}{a} \frac{da}{dT} \]
\[ \alpha_V = \frac{1}{V} \frac{dV}{dT} = \frac{1}{\frac{a^3}{3}} \frac{d\left(a^3\right)}{dT} = \frac{1}{a^3} 3a^2 \frac{da}{dT} = \frac{3}{a} \frac{da}{dT} \]
\[ \alpha_L = 3\alpha_L \]

A.3 Specific Heat

\[ dU = \delta Q + \delta W \]
\[ = \delta Q - PdV \]
Let \( dV = 0 \)
\[ \frac{\partial U}{\partial T} = \left( \frac{\partial Q}{\partial T} \right)_V \]
\[ \left( \frac{\partial Q}{\partial T} \right)_V = C_V \]
\[ H = U + PV \]
\[ dH = \delta Q + V \, dP \]

Let \( dP = 0 \)
\[
\frac{\partial H}{\partial T} = \left( \frac{\partial Q}{\partial T} \right)_P
\]
\[
\left( \frac{\partial Q}{\partial T} \right)_P = C_P
\]
Appendix B

Simulation Scripts

B.1 General System Setup

Listing B.1: Script to build the simulation system

```python
#!/usr/bin/env python
from numpy import *
from random import randint
from random import choice
from random import random
from random import seed
from cPickle import dump
from argparse import ArgumentParser
from scipy.integrate import trapz
from scipy.interpolate import interp1d

# Parser Setup for Command-line Arguments

parser = ArgumentParser(description="Preprocessor for LAMMPS simulations")

parser.add_argument('-coeffs',default='gen.tersoff',help='Path to coefficients file')

parser.add_argument('-R0',default=4,type=int,help='Unit cells in x')
parser.add_argument('-R1',default=4,type=int,help='Unit cells in y')
parser.add_argument('-R2',default=4,type=int,help='Unit cells in z')

parser.add_argument('-a',default=5.6535,type=float,help='Lattice parameter in angstroms')

parser.add_argument('-QCL',default=False,help='Force superlattice structure',action='store_true')
parser.add_argument('-QCL2',default=False,help='Force superlattice structure 2',action='store_true')
parser.add_argument('-III',default='In',help='Group III type for III-V semiconductor')

parser.add_argument('-dt',default='1',type=float,help='Time step in fs')
parser.add_argument('-dt_dump',default='40',type=float,help='Dump time step in fs')

parser.add_argument('-t_relax',default='10',type=float,help='System relaxation time in ps')
parser.add_argument('-t_equil',default='10',type=float,help='System equilibration time in ps')
parser.add_argument('-t_sim',default='40',type=float,help='System simulation time in ps')

parser.add_argument('-seed',default=None,type=int,help='Random number seed for lattice generation')

# Poke some of the parser inputs
```
P = vars(parser.parse_args())
if P['QCL']:
P['R0'] = 78
P['III'] = 'QCL'
if P['QCL2']:
P['R0'] = 227
P['III'] = 'QCL2'
seed(P['seed'])
P['dt'] = P['dt']/1000
P['dt_dump'] = P['dt_dump']/1000
# Superlattice setup#
# Superlattice data
# Pasted from spreadsheet
S = ""
S = ""
S2 = ""
S2 = ""
```python
# Number of points used in integration
N = 1000

def dist(f, s0):
def fromText(S):
    F = array(map(float, S.split())).reshape([-1, 2])
    return F[:, 0], F[:, 1]

def discrete(S, N):
    L, T = fromText(S)
    x = linspace(0, L.sum(), N)
    G = empty([N])
    for k in xrange(L.size):
        xl = L[:k].sum()
        xh = L[:k+1].sum()
        kl = argmin(abs(x-xl))
        kh = argmin(abs(x-xh))
        G[kl:kh+1] = T[k]
    return x, G

def discrete(s0, N):
x, G = discrete(s0, N)
p = empty([N])
for k in xrange(N):
    pp = trapz(G*f(x-x[k]-x[-1]), x)
    pc = trapz(G*f(x-x[k]), x)
    pn = trapz(G*f(x-x[k]+x[-1]), x)
    p[k] = pp+pc+pn
pf = interp1d(x, p, bounds_error=False, fill_value=0.5)
return pf

# Selector for superlattice
l = P['a']/4
p = dist(lambda x: exp(-0.5*(x/l)**2)/(l*sqrt(2*pi)), S)
p2= dist(lambda x: exp(-0.5*(x/l)**2)/(l*sqrt(2*pi)), S2)

def superlattice(x):
r1 = random()
r2 = random()
s = p(x[0])
o = 0
if r1<s and r2>0.52:
o = 2
elif r1<s and r2<0.52:
```
```python
o = 3
elif r1>=s and r2>0.53:
o = 1
elif r1>=s and r2<=0.53:
o = 3
if o==0:
    print r1,r2,s,x[0]
return o

def superlattice2(x):
    r1 = random()
    r2 = random()
    s = p2(x[0])
    if r1>=s:
        return 2
    elif r1<s and r2>0.15:
        return 2
    elif r1<s and r2<=0.15:
        return 1

# Lattice Characteristics#

# Lattice Parameter
a = P['a']

# Number unit cells
R = array([P['R0'],P['R1'],P['R2']])

# Atom Types
Al = lambda x: 1
Ga = lambda x: 2
In = lambda x: 3
As = lambda x: 4

AlGa = lambda x: choice([1,2])
GaAl = lambda x: choice([1,2])
AlIn = lambda x: choice([1,3])
InAl = lambda x: choice([1,3])
GaIn = lambda x: choice([2,3])
InGa = lambda x: choice([2,3])

QCL = lambda x: superlattice(x)
QCL2 = lambda x: superlattice2(x)

# [Al, Ga, In, As]
M = array([26.9815386, 69.723, 114.818, 74.92159])

# =====================#
# Simulation Settings#

Sim = {}

Sim['P00'] = 0.0
Sim['T00'] = 300.0
Sim['T0'] = 2*Sim['T00']

Sim['dt'] = P['dt']
Sim['dt_dump'] = P['dt_dump']
Sim['t'] = 'step*$dt'
Sim['tau_T'] = 100*Sim['dt']
Sim['tau_P'] = 1000*Sim['dt']

Sim['steps_relax'] = int(1.0/Sim['dt']*P['t_relax'])
Sim['steps_equil'] = int(1.0/Sim['dt']*P['t_equil'])
Sim['steps_sim'] = int(1.0/Sim['dt']*P['t_sim'])
Sim['steps_dump'] = int(Sim['dt_dump']/Sim['dt'])
Sim['steps_thermo'] = int(0.1/Sim['dt'])

Sim['a'] = a
for k in range(R.size):
    Sim['R%d'%k] = R[k]
for k in range(M.size):
    Sim['M%d'%k] = M[k]

Sim['seed'] = randint(100000,1000000)

# Atom Writer

fcc = array([[0,0,0],[.5,.5,0],[.5,0,.5],[0,.5,.5]])
atom_idx = 1

def writeLattice(f,o,t):
global atom_idx
for k in xrange(R[2]):
    for j in xrange(R[1]):
        for i in xrange(R[0]):
            for e in fcc:
                x = a*(e+array([i,j,k],dtype=double))+a*o
                f.write('%d␣%d␣%f␣%f␣%f
'%(atom_idx,t(x),x[0],x[1],x[2]))
            atom_idx += 1

# Write Diamond Structure

f = open('struct.xyz','w')
f.write('#␣Input␣geometry␣for␣lammps
Na = 2*fcc.shape[0]*product(R)
f.write('%d atoms
' % Na)
f.write('%d atom types
' % M.size)
for k in [0, 1, 2]:
    f.write('%f %f %slo %shi
' % (0.0, a*R[k], 'xyz'[k], 'xyz'[k]))
f.write('
')
f.write('Masses

')
for k in range(M.size):
    f.write('%d %f
' % (k+1, M[k]))
f.write('
')
f.write('Atoms

')
writeLattice(f, 0.00*ones([3]), E_V )
writeLattice(f, 0.25*ones([3]), E_IIV)

f.close()

# Write Variable Includes

f = open('vars.inc','w')
for k in sorted(Sim.keys()):
    f.write('variable %s equal %s
' % (k, Sim[k].__str__()))
    f.write('variable coeffs string %s' % P['coeffs'])
f.close()

Sim['coeffs'] = P['coeffs']
f = open('vars.pk','w')
dump(Sim,f)
f.close()
Listing B.2: Script to construct the potential coefficient file

```python
#!/usr/bin/env python
from numpy import *
from sys import argv

names = "m␣g␣l3␣c␣d␣h␣n␣b␣l2␣B␣R␣D␣l1␣A".split()

lit_values = "Al-Ga 3␣1␣1.5␣0.075563␣19.268250␣3.243333␣4.597590␣0.273372␣1.209135␣55.9917␣3.5␣0.1␣2.54688␣699.824
Al-In 3␣1␣1.5␣0.074836␣19.569127␣-0.659266␣6.085605␣0.316846␣1.304306␣91.1331503975␣3.5␣0.1␣2.54688␣699.824
In-Al 3␣1␣1.5␣0.084215␣19.262600␣7.392280␣3.402230␣2.108710␣1.304306␣91.1331503975␣3.5␣0.1␣2.600618␣699.824
In-Ga 3␣1␣1.5␣0.080159␣19.527700␣7.268050␣3.437390␣0.705241␣1.583140␣221.557000␣3.5␣0.1␣2.561600␣1719.70000
Al-As 3␣1␣1.5␣1.449752␣0.828713␣-0.520946␣4.047579␣0.330946␣1.558230␣219.139600␣3.5␣0.1␣2.809295␣2307.879400
Ga-As 3␣1␣1.5␣1.226302␣0.790396␣0.518489␣6.317410␣0.357192␣1.723012␣314.459660␣3.5␣0.1␣2.561600␣1719.70000
In-As 3␣1␣1.5␣5.172421␣1.665967␣0.541331␣0.756169␣0.318640␣1.422429␣266.571631␣3.6␣0.1␣2.597556␣1968.295443
Al-Al 3␣1␣1.5␣0.074836␣19.569127␣-0.659266␣6.085605␣0.316846␣0.927442␣23.031117␣3.5␣0.1␣2.585337␣492.765900
Ga-Ga 3␣1␣1.5␣0.076298␣19.796474␣7.145917␣3.472041␣0.339046␣1.490824␣136.123032␣3.5␣0.1␣2.508427␣933.888094
In-In 3␣1␣1.5␣0.084215␣19.262600␣7.392280␣3.402230␣2.108710␣1.681170␣360.610000␣3.6␣0.1␣2.615900␣2975.540000
As-As 3␣1␣1.5␣5.273131␣0.750262␣0.518489␣6.317410␣0.357192␣1.723012␣314.459660␣3.6␣0.1␣2.384132␣1571.860840
""

lit_values_unsure = "Al-Ga 3.000000␣1.000000␣1.500000␣0.075563␣19.268250␣3.243333␣4.597590␣0.273372␣1.209135␣55.9917␣3.500000␣0.100000␣2.54688␣699.824
Al-In 3.000000␣1.000000␣1.500000␣0.079526␣19.415864␣3.365680␣4.744368␣0.727667␣1.304306␣91.133150␣3.500000␣0.100000␣2.600618␣699.824
In-Ga 3.000000␣1.000000␣1.500000␣0.080159␣19.527700␣7.268050␣3.437390␣0.705241␣1.583140␣221.557000␣3.500000␣0.100000␣2.561600␣1719.70000
Al-As 3.000000␣1.000000␣1.500000␣1.449752␣0.828713␣-0.520946␣4.047579␣0.330946␣1.558230␣219.139600␣3.500000␣0.100000␣2.809295␣2307.879400
Ga-As 3.000000␣1.000000␣1.500000␣1.226302␣0.790396␣0.518489␣6.317410␣0.357192␣1.723012␣314.459660␣3.500000␣0.100000␣2.561600␣1719.70000
In-As 3.000000␣1.000000␣1.500000␣5.172421␣1.665967␣0.541331␣0.756169␣0.318640␣1.422429␣266.571631␣3.600000␣0.100000␣2.597556␣1968.295443
Al-Al 3.000000␣1.000000␣1.500000␣0.074836␣19.569127␣-0.659266␣6.085605␣0.316846␣0.927442␣23.031117␣3.500000␣0.100000␣2.585337␣492.765900
Ga-Ga 3.000000␣1.000000␣1.500000␣0.076298␣19.796474␣7.145917␣3.472041␣0.339046␣1.490824␣136.123032␣3.500000␣0.100000␣2.508427␣933.888094
In-In 3.000000␣1.000000␣1.500000␣0.084215␣19.262600␣7.392280␣3.402230␣2.108710␣1.681170␣360.610000␣3.600000␣0.100000␣2.615900␣2975.540000
As-As 3.000000␣1.000000␣1.500000␣5.273131␣0.750262␣0.518489␣6.317410␣0.357192␣1.723012␣314.459660␣3.600000␣0.100000␣2.384132␣1571.860840
""

lit_values_unsure = """
```

def transform(values):
    D = {}
    for r in values.split('
'):
        if r=='' or r.startswith('#'):
            continue
        w = r.split('	')
        key = tuple(w[0].split('-'))
        vals = w[1:]
        D[key] = {}
        for k in xrange(len(names)):
            D[key][names[k]] = vals[k]
    return D

def atoms(D):
    S = set()
    for k in D:
        S.add(k[0])
        S.add(k[1])
    return list(S)

def expand(D):
    S = atoms(D)
    for i in S:
        for j in S:
            if (not (i,j) in D) and ((j,i) in D):
                D[(i,j)] = D[(j,i)]
    return D

def poke(D,p,c,v):
    D[(p[0],p[1])][c] = v

def rpoke(D,p,c,v):
    D[(p[0],p[1])][c] = '%s'%(v*float(D[(p[0],p[1])][c]))

def save(fn,D):
    f = open(fn,'w')
    S = atoms(D)
    for k in D:
        for s in S:
            L = [k[0],k[1],s]
            for n in names:
                L.append(D[k][n])
            for l in L:
                f.write(l.__str__()+'\t')
            f.write('
')
    f.close()

def lit():
    return expand(transform(lit_values))

if __name__=='__main__':
    D = transform(lit_values)
    if len(argv)==1 and False:
print 'Usage:\n\t{III}-{IV}:{coefficient}\%={value}'

for a in argv[1:]:
k,v = a.split('=')
p,c = k.split(':')
mode = poke
if c[-1]=='\%':
c = c[:-1]
mode = rpoke
p = tuple(p.split('-'))
v = float(v)
mode(D,p,c,v)

F = expand(D)
save('gen.tersoff',F)
B.2 Thermal Conductivity Calculation

Listing B.3: Thermal conductivity job script

```bash
#!/bin/bash

PBS -N CONDUCTIVITY
PBS -l nodes=1:ppn=24
PBS -l walltime=48:00:00
PBS -q batch
PBS -M horne.kyle@gmail.com
PBS -m a
PBS -t 1-50

BASE_DIR=${DORC_HOME}/Conductivity
WORK_DIR=${BASE_DIR}/run_${PBS_ARRAYID}

if [-d "${WORK_DIR}" ]; then
    rm -rf ${WORK_DIR}
fi
mkdir ${WORK_DIR}
cd ${WORK_DIR}

TIMES="-t_Relax 50 -t_equil 200 -t_sim 400"
SIZES="-R0 78 -R1 6 -R2 6"

../generate.py
../build.py ${TIMES} ${SIZES} -III AlIn
mpirun -np ${PBS_NP} lammps -in ../TC.lmp
```
Listing B.4: Thermal conductivity lammps script

```plaintext
# Input file for Green-Kubo thermal conductivity#
#================================================#

units metal
atom_style atomic
boundary p p p

include vars.inc
variable T equal temp
variable V equal vol
variable PE equal pe
variable KE equal ke
read_data struct.xyz

group Al type 1
group Ga type 2
group In type 3
group As type 4

pair_style tersoff
pair_coeff * * {coeffs} Al Ga In As

neighbor 2 bin
neigh_modify delay 5
velocity all create {T0} {seed} rot yes sun yes mom yes dist gaussian loop geom
timestep {dt}

#===========================================#
# Setup log output #
#===========================================#

thermo ${steps_thermo}
thermo_style custom v_t temp etotal ke pe

#=================================#
# Relax the lattice to the potential in NTP #
#=================================#

fix NPT all npt temp {T00} {T00} {tau_T} iso {P00} {P00} {tau_P}
reset_timestep 0
run ${steps_relax}
unfix NPT

#=================================#
# Equilibriate the system in NVE #
#=================================#

fix ENERGY_OUT all print {steps_thermo} "(t) (T) (KE) (PE)" &
title "# Time, Temp, KE, PE" screen no file E.dat
```
fix NVE all nve temp ${T00} ${T00} ${tau_T}
reset_timestep 0
run ${steps_equil}
unfix ENERGY_OUT

# Simulate system and compute flux in NVE #
#=========================================#

compute myTemp all temp
compute myKE all ke/atom
compute myPE all pe/atom
compute myStress all stress/atom virial
compute VJ all heat/flux myKE myPE myStress

variable Jx equal (c_VJ[1]-c_VJ[4])/$V$
variable Jy equal (c_VJ[2]-c_VJ[5])/$V$
variable Jz equal (c_VJ[3]-c_VJ[6])/$V$

fix FLUX_OUT all print ${steps_dump} "$t" "$T" $Jx $Jy $Jz" &
title "# Time, Temp, Jx, Jy, Jz" screen no file J.dat

reset_timestep 0
thermo ${steps_dump}
thermo_style custom v_t temp etotal v_Jx v_Jy v_Jz
run ${steps_sim}

#=======================================================#
# Write volume out to a file for Green-Kubo calculation #
#=======================================================#

shell echo $V > vol.dat
Listing B.5: Thermal conductivity post-processing script

```python
#!/usr/bin/env python
from numpy import *
from scipy.io import savemat
from scipy.integrate import cumtrapz
import os
import os.path
import sys

def smooth(x,N):
    y = zeros(len(x))
    for k in range(len(x)):
        s = x[max(0,k-N):min(len(x),k+N+1)]
        y[k] = sum(s)/len(s)
    return y

def autocorr(x):
    AC = correlate(x,x,'full')[-x.size:]
    for k in xrange(AC.size):
        AC[k] = AC[k]/(x.size-k)
    return AC

def getRun(fn):
    # print fn
    V = loadtxt(fn+'/vol.dat')
    t,T,Jx,Jy,Jz = loadtxt(fn+'/J.dat',unpack=True)
    t = t-t[0]
    JJx = autocorr(Jx)
    JJy = autocorr(Jy)
    JJz = autocorr(Jz)
    #A = ( JJx + JJy + JJz ) /3
    #A = ( JJy + JJz ) /2
    A = JJx
    I = zeros(A.size)
    I[1:] = cumtrapz(A,t)
    t2,T2,K,P = loadtxt(fn+'/E.dat',unpack=True)
    return V,t,T,A,I,K,P,JJx,JJy,JJz

runs = []
files = os.listdir`
for f in files:
    if(not f.startswith('run-GaIn_')):
        continue
    if(not os.path.exists(f+'/J.dat')):
        continue
    if(not os.path.exists(f+'/vol.dat')):
        continue
    runs.append(f)
print 'Successful Run Count:',len(runs)
```

D = {}
V,t,T,A,I,K,P,JJx,JJy,JJz = getRun(runs[0])
D['V'] = zeros([len(runs)])
D['t'] = zeros([t.size,len(runs)])
D['T'] = zeros([T.size,len(runs)])
D['A'] = zeros([A.size,len(runs)])
D['I'] = zeros([I.size,len(runs)])
D['K'] = zeros([K.size,len(runs)])
D['P'] = zeros([P.size,len(runs)])
D['JJx'] = zeros([JJx.size,len(runs)])
D['JJy'] = zeros([JJy.size,len(runs)])
D['JJz'] = zeros([JJz.size,len(runs)])

print 'Combining...' 
for k in xrange(len(runs)):
    p = 100.0*k/len(runs)
    sys.stdout.write('[=%s]␣%5.1f%%'%'='*int(p/5),'␣'*(20-int(p/5)),p)
    sys.stdout.flush()
    Vl,tl,tl,kl,kl,pl,JJxl,JJyl,JJzl = getRun(runs[k])
    D['V'][:,k] = Vl
    D['t'][:,k] = tl
    D['T'][:,k] = Tl
    D['A'][:,k] = Al
    D['I'][:,k] = Il
    D['K'][:,k] = Kl
    D['P'][:,k] = Pl
    D['JJx'][:,k] = JJxl
    D['JJy'][:,k] = JJyl
    D['JJz'][:,k] = JJzl
    p = 100
    sys.stdout.write('[=%s]␣%5.1f%%
'%'='*int(p/5),'␣'*(20-int(p/5)),p)
    sys.stdout.flush()

savemat('combined.mat',D)
Listing B.6: Thermal conductivity plotting script

```python
#!/usr/bin/env python

import matplotlib as mpl
mpl.rcParams['font.family'] = 'sans'
mpl.rcParams['font.size'] = 11
mpl.rcParams['font.serif'] = 'Times New Roman'
mpl.rcParams['image.cmap'] = 'coolwarm'
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['path.simplify'] = True
mpl.rcParams['path.simplify_threshold'] = 0.5
mpl.rcParams['figure.figsize'] = 5.3,4

from pylab import *rom sys import argv
from scipy.io import loadmat
from scipy.optimize import curve_fit
from scipy.optimize import leastsq

f = lambda t,A,t0,p: A*(1-exp(-(t/t0)**p))
fp = lambda t,A,t0,p: A*p*t**(p-1)*t0**(-1)*exp(-(t/t0)**p)
def fit(t,I):
    weights = linspace(1,10,t.size)
    DG = array([1.0E-7,5.0,1.0])
    D0 = [1.0E-6,50.0,1.0]
    cI,vI = curve_fit(f,t,I,p0=D0,diag=DG,maxfev=100000,sigma=weights)
    It = f(t,cI[0],cI[1],cI[2])
    return cI[0],cI[1],cI[2] # A, t, p

af = lambda t,A0,A1,t0,t1: A0*(1-exp(-t/t0))+A1*(1-exp(-t/t1))
afp = lambda t,A0,A1,t0,t1: A0/t0*exp(-t/t0)+A1/t1*exp(-t/t1)
def fit2(t,I):
    def res(p,t,I,w):
        A0,A1,t0,t1 = p
        rss = ((I-af(t,A0,A1,t0,t1))/w)**2
        return rss
    weights = linspace(1,10,t.size)
    D0 = [1.0E-8,1.0E-6,0.05,50.0]
    #cI , vI = curve_fit (af ,t ,I , p0 =D0 , diag =DG , maxfev =100000 , sigma =weights )
    cI,vI = leastsq(res,args=(t,I,weights),x0=D0)
    It = af(t,cI[0],cI[1],cI[2],cI[3])
    return cI[0],cI[1],cI[2],cI[3] # A0 , A1 , t0 , t1

def conductivity(V,T,FI):
    kB = 8.6173332478E-5
    convert = 1602.177
    k = V/(kB*T**2)*FI*convert
    return k

def read_file(fn):
    D = loadmat(fn)
    for k in D.keys():
        exec("%s = D['%s']"%(k,k))
    R = {}
```
tm = t.mean(1)
k = argmin(abs(tm-1000))
tm = tm[:kk]
R[ 't' ] = tm
R[ 'rt' ] = t[:kk]
R[ 'V' ] = V[:kk]
R[ 'T' ] = T.mean(1)[:kk]
R[ 'uT' ] = T.std(1)[:kk]/sqrt(N)
R[ 'A' ] = A.mean(1)[:kk]
R[ 'rA' ] = A[:kk]
R[ 'uA' ] = A.std(1)[:kk]/sqrt(N)
R[ 'I' ] = I.mean(1)[:kk]
R[ 'rI' ] = I[:kk]
R[ 'uI' ] = I.std(1)[:kk]/sqrt(N)
R[ 'K' ] = K.mean(1)[:kk]
R[ 'uK' ] = K.std(1)[:kk]/sqrt(N)
R[ 'P' ] = P.mean(1)[:kk]
R[ 'uP' ] = P.std(1)[:kk]/sqrt(N)
return R

def fill_between(x, y1, y2=0, ax=None, **kwargs):
    ax = ax if ax is not None else plt.gca()
    ax.fill_between(x, y1, y2, **kwargs)
p = plt.Rectangle((0, 0), 0, 0, **kwargs)
    ax.add_patch(p)
    return p

fn = argv[1]
print('=␣%s␣======================================='%pf)

R = read_file(fn)
t = R[ 't' ]
A = R[ 'A' ]
uA = R['uA']
I = R[ 'I' ]
uI = R['uI']
V = R[ 'V' ].mean()
T = R[ 'T' ].mean()
A0,t0,p0 = fit(t,I)
fI = f(t,A0,t0,p0)
fA = fp(t,A0,t0,p0)
k = conductivity(V,T,A0)
print('A␣␣␣=␣%12g␣[eV^2␣*␣ps^(-1)␣*␣AA^(-4)]'%A0
print('tau␣=␣%12g␣[ps]'%t0
print('p0␣␣=␣%12g␣[1]'%p0
print('k␣␣␣=␣%12g␣[W␣*␣m^(-1)␣*␣K^(-1)]'%k
print('')
aA0,aAt0,aAt1 = fit2(t,I)
```
afI = af(t,aA0,aA1,at0,at1)
afA = afp(t,aA0,aA1,at0,at1)
ak = conductivity(V,T,aA0+aA1)

print 'A = (%12g %12g) [eV^2 * ps^{-1} * AA^{-4}] '%(aA0,aA1)
print 'tau = (%12g %12g) [ps]'%(at0,at1)
print 'k = %12g [W * m^{-1} * K^{-1}] '%ak

# quit ()
#S = abs ( rfft (A))
#w = 2* pi * linspace (0 ,0.5/(1.0 e -12*( t [1] - t [0]))) ,S. size )
# print S [0]
# print A0 ,S [0]/ A0
# print I [-1] , S [0]/ I [-1]
# print S [0]* t [ -1]/ t. size
# fig = figure (' Spectral Conductivity ')
# ax = subplot (111)
# plot (w /(2* pi ) /1 e12 ,S*t [ -1]/ t. size ,'r-'
# semilogy ()
# ylabel (r \int_0 ^t \left <\vec {J} \left (0\right )\cdot \vec {J} \left (t\right )\right >e ^{i w t} dt
# $\[ eV ^2 ps ^{-1} \AA ^{-4} \] ')
# xlabel (r ' Frequency $f$ [THz ] ')
# tight_layout ()

kt0 = argmin(abs(t-t0))

fig = figure('Autocorrelation␣Hexbin')
ax = subplot(111)
hexbin(R['rt'].flatten(),R['rA'].flatten(),gridsize=50,extent=(0,2*t0,-10*uA[-1],10*uA[-1]))
plot(t[1:2*kt0],fA[1:2*kt0],'r-',lw=2,alpha=0.7,label='Fit␣1')
plot(t[1:2*kt0],afA[1:2*kt0],'m-',lw=2,alpha=0.7,label='Fit␣2')
xlim(0,2*t0)
ylim(-10*uA[-1],10*uA[-1])
ax.ticklabel_format(axis='y', style='sci', scilimits=(-1,1))
xlabel(r'Time $t$ [ps]')
ylabel(r'$\int_0^t\langle \vec {J} (0) \cdot \vec {J} (t) \rangle dt$ [eV^2 ps^{-1} \AA^{-4}]')
legend(loc='best',fancybox=True)
tight_layout()
savefig('%s-autocorrelation-hexbin.svg'%pf)

fig = figure('Autocorrelation␣Lines')
ax = subplot(111)
plot(R['rt'][:,0],R['rA'][:,0],'b-',lw=0.5,alpha=0.2,label='Single␣Trajectories')
for i in xrange(1,R['rt'].shape[1]):
    plot(R['rt'][:,i],R['rA'][:,i],'b-',lw=0.5,alpha=0.2)
plot(t,A,'r-',lw=2,alpha=0.7,label='Mean␣Trajectory')
ax.ticklabel_format(axis='y', style='sci', scilimits=(-1,1))
xlabel(r'Time $t$ [ps]')
ylabel(r'$\int_0^t\langle \vec {J} (0) \cdot \vec {J} (t) \rangle dt$ [eV^2 ps^{-1} \AA^{-4}]')
legend(loc='best',fancybox=True)
tight_layout()
savefig('%s-autocorrelation-lines.svg'%pf)

fig = figure('Autocorrelation')
```
ax = subplot(111)
s = 50
fill_between(t[:,s]-uA[:,s],A[:,s]+uA[:,s],label='Uncertainty',alpha=0.3, lw=0)
plot(t, A, 'b-', label='Simulation', lw=2)
plot(t, fA, 'r-', label='Fit\(1\): \(\tau = %.1f \text{ps}\)'%t0, lw=2, alpha=0.7)
plot(t, afA, 'm-', label='Fit\(2\): \((%.1f %.1f) \text{ps}\)'%(at0, at1), lw=2, alpha=0.7)
ylim(-A.std(), max(A.std(), fA[1]))
ax.ticklabel_format(axis='y', style='sci', scilimits=(-1,1))
xlabel(r'\(t\text{[ps]}\)')
ylabel(r'\(\langle \vec{J}(0) \cdot \vec{J}(t) \rangle \text{[eV}^2 \text{ps}^{-1} \text{Å}^{-4}]\)')
legend(loc='best', fancybox=True)
tight_layout()
savefig('%s-autocorrelation.svg'%pf)

fig = figure('Integral Hexbin')
ax = subplot(111)
hexbin(R['rt'].flatten(), R['rI'].flatten(), gridsize=50, extent=(0, 2*t0, -uI[-1], I[-1]+uI[-1]))
plot(t[1:2*kt0], fI[1:2*kt0], 'r-', lw=2, alpha=0.7, label='Fit\(1\)')
plot(t[1:2*kt0], afI[1:2*kt0], 'm-', lw=2, alpha=0.7, label='Fit\(2\)')
xlim(0, 2*t0)
xlabel(r'\(t\text{[ps]}\)')
ylabel(r'\(\int_0^t \langle \vec{J}(0) \cdot \vec{J}(t) \rangle dt \text{[eV}^2 \text{ps}^{-1} \text{Å}^{-4}]\)')
legend(loc='best', fancybox=True)
tight_layout()
savefig('%s-integral-hexbin.svg'%pf)

fig = figure('Integral Lines')
ax = subplot(111)
plot(R['rt'][:,0], R['rI'][:,0], 'b-', lw=0.5, alpha=0.2, label='Single Trajectories')
for i in xrange(1, R['rt'].shape[1]):
    plot(R['rt'][:,i], R['rI'][:,i], 'b-', lw=0.5, alpha=0.2)
plot(t, I, 'r-', lw=2, alpha=0.7, label='Mean Trajectory')
xlabel(r'\(t\text{[ps]}\)')
ylabel(r'\(\int_0^t \langle \vec{J}(0) \cdot \vec{J}(t) \rangle dt \text{[eV}^2 \text{ps}^{-1} \text{Å}^{-4}]\)')
legend(loc='best', fancybox=True)
tight_layout()
savefig('%s-integral-lines.svg'%pf)

fig = figure('Integral')
ax = subplot(111)
s = 50
fill_between(t[:,s]-uI[:,s], I[:,s]+uI[:,s], label='Uncertainty', alpha=0.3, lw=0)
plot(t, I, 'b-', label='Simulation', lw=2)
plot(t, fI, 'r-', label='Fit\(1\): \(k = %.1f \text{W/m} \cdot \text{K}\)'%k, lw=2, alpha=0.7)
plot(t, afI, 'm-', label='Fit\(2\): \(k = %.1f \text{W/m} \cdot \text{K}\)'%ak, lw=2, alpha=0.7)
ax.ticklabel_format(axis='y', style='sci', scilimits=(-1,1))
xlabel(r'\(t\text{[ps]}\)')
ylabel(r'\(\int_0^t \langle \vec{J}(0) \cdot \vec{J}(t) \rangle dt \text{[eV}^2 \text{ps}^{-1} \text{Å}^{-4}]\)')
legend(loc='best', fancybox=True)
tight_layout()
savefig('%s-integral.svg'%pf)

# show()
B.3 Dispersion Curve Calculation

Listing B.7: Dispersion curve job script

```bash
#!/bin/bash

#PBS -N CONDUCTIVITY
#PBS -N CONDUCTIVITY
#PBS -l nodes=1:ppn=8
#PBS -l walltime=4:00:00
#PBS -q short
#PBS -M horne.kyle@gmail.com
#PBS -m a
#PBS -t 1-1

BASE_DIR=$DORC_HOME/DisperseConductivity
WORK_DIR=$BASE_DIR/run_${PBS_ARRAYID}

TIMES="-t_relax 5 -t_equil 20 -t_sim 40 -dt_dump 15"
R=8
SIZES="-R0 $R -R1 $R -R2 $R"

if [ -d "$WORK_DIR" ]; then
  rm -rf $WORK_DIR
fi
mkdir $WORK_DIR
cd $WORK_DIR

../generate.py
../build.py $TIMES $SIZES -QCL
mpirun -np $PBS_NP lammps -in ../DISPTC.lmp

R0=$(grep R0 vars.inc | awk '{print $4}')
dt=$(grep dt vars.inc | awk '{print $4}')
../bin2nc -s ixv.bin
../qcl lammps.nc $R0 $dt
../plot.py L.nc
```
#============================================#
# Input file for dispersion and conductivity  #
#============================================#

units metal
atom_style atomic
boundary p p p

include vars.inc
variable T equal temp
variable V equal vol
variable PE equal pe
variable KE equal ke
read_data struct.xyz

group Al type 1
group Ga type 2
group In type 3
group As type 4

pair_style tersoff
pair_coeff * * {coeffs} Al Ga In As

neighbor 2 bin
neigh_modify delay 5
velocity all create {T0} {seed} rot yes sun yes mom yes dist gaussian loop geom

timestep $(dt)

#===========================================
# Relax the lattice to the potential in NTP  #
#===========================================

fix NPT all npt temp {T00} {T00} {tau_T} iso {P00} {P00} {tau_P}
reset_timestep 0
run $(steps_relax)
unfix NPT

#===========================================
# Equilibriate the system in NVE           #
#===========================================

fix ENERGY_OUT all print $(steps_thermo) "$(T) $(T) $(KE) $(PE)" & title "# Time, Temp, KE, PE" screen no file E.dat
fix NVE all nve temp $(T00)$ $(T00)$ $(tau_T)$
reset_timestep 0
run $(steps_equil)$
unfix ENERGY_OUT

#=========================================#
# Simulate system and compute flux in NVE #
#=========================================#

compute myTemp all temp
compute myKE all ke/atom
compute myPE all pe/atom
compute myStress all stress/atom myTemp virial
  # xx xy xz 1 4 5
  # yx yy yz (4) 2 6
  # zx zy zz (5)(6) 3
compute VJ all heat/flux myKE myPE myStress

variable Jx equal (c_VJ[1]-c_VJ[4])/$V$
variable Jy equal (c_VJ[2]-c_VJ[5])/$V$
variable Jz equal (c_VJ[3]-c_VJ[6])/$V$

variable qx atom (c_myKE+c_myPE)*vx+(c_myStress[1]*vx+c_myStress[4]*vy+c_myStress[5]*vz)
variable qy atom (c_myKE+c_myPE)*vy+(c_myStress[4]*vx+c_myStress[2]*vy+c_myStress[6]*vz)
variable qz atom (c_myKE+c_myPE)*vz+(c_myStress[5]*vx+c_myStress[6]*vy+c_myStress[3]*vz)

fix FLUX_OUT all print $(steps_dump) "$T" "$Jx" "$Jy" "$Jz" &
title "# Time, Temp, Jx, Jy, Jz" screen no file J.dat

dump VELX_OUT all custom $(steps_dump) ixv.bin id type mass x y z vx vy vz v_qx v_qy v_qz
reset_timestep 0
thermo $(steps_dump)
thermo_style custom v_t temp v_Jx v_Jy v_Jz
run $(steps_sim)

#=======================================================#
# Write volume out to a file for Green-Kubo calculation #
#=======================================================#

shell echo $(V) > vol.dat
Listing B.9: Dispersion curve plotting script

```python
#!/usr/bin/env python
import matplotlib as mpl
mpl.use('Agg')
mpl.rcParams['font.family'] = 'sans'
mpl.rcParams['font.size'] = 11
mpl.rcParams['font.serif'] = 'Times New Roman'
mpl.rcParams['image.cmap'] = 'coolwarm'
mpl.rcParams['figure.figsize'] = 4, 2.8
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['path.simplify'] = True
mpl.rcParams['path.simplify_threshold'] = 1.0
from pylab import *
from sys import argv
from matplotlib.gridspec import GridSpec
from scipy.io import netcdf_file
from h5py import File as hdf5_file
import os.path

fn = argv[1]
pf = fn.split('.')[0]
ef = '../Experiments/%s.h5'%pf

have_exp = False
if os.path.isfile(ef):
    H = hdf5_file(ef)
    have_exp = True

N = netcdf_file(fn)
f = N.variables['f'][:]/1.0E12

def smooth(x,N):
k = linspace(-2,2,2*N+1)
w = exp(-k**2)
return correlate(x,w,mode='same')

def mirror(a):
    return a[:,::-1]

def subplots(fig):
gs = GridSpec(1,4,width_ratios=[1, sqrt(2.0), sqrt(3.0)/2.0, sqrt(2.0)])
ax = []
ax.append(fig.add_subplot(gs[0]))
ax.append(fig.add_subplot(gs[1], sharey=ax[0]))
ax.append(fig.add_subplot(gs[2], sharey=ax[0]))
ax.append(fig.add_subplot(gs[3], sharey=ax[0]))

ax[0].set_ylabel(r'Frequency $f$ [THz]')
setp(ax[1].get_yticklabels(), visible=False)
setp(ax[2].get_yticklabels(), visible=False)
setp(ax[3].get_yticklabels(), visible=False)

ax[0].set_xticks([0.0, 0.5])
setp(ax[0].set_xticklabels(['$\Gamma$', '$\Delta_c$']))
```
def curve(ax,W,m):
    L = []
    for r in W:
        for c in r:
            L.append(c)
    A = None
    for n in L:
        V = empty(N.variables[n].data.shape)
        V[:,1] = N.variables[n].data[:,1]
        for l in xrange(V.shape[1]):
            V[:,l] = log(V[:,l]/median(V[:,l]))
        if m:
            V = mirror(V)
        if A==None:
            A = V
        else:
            A = maximum(A,V)
    k = linspace(0,1,A.shape[1])
    ax.pcolormesh(k,f,A,rasterized=True,shading='gouraud')

def dos(ax,m):
    DOS = N.variables['DOS'].data[:]
    sDOS = smooth(DOS,3)
    sDOS /= sDOS.max()
    ax.plot(sDOS,f,'r-',lw=2,alpha=0.75)
    if m<2:
        ax.plot(sDOS,2*f,'r:',lw=2,alpha=0.75)
    SPK = N.variables['SPK'].data[:]
    sSPK = smooth(SPK,3)
    sSPK /= sSPK.max()
    ax.plot(sSPK,f,'b-',lw=2,alpha=0.75)
    ax.set_xticklabels(['DOS/SPQ'])
    ax.set_ylim(0,f.max()/m)

b = ['L0','T1','T2']
fig = figure()
ax = subplot(fig)
curve(ax[0],[[b[i]+'pGpX',b[i]+'sGpX'] for i in xrange(len(b))],False)
curve(ax[1],[[b[i]+'sXpG',b[i]+'sXsG'] for i in xrange(len(b))],False)
curve(ax[2],[[b[i]+'pGpL',b[i]+'tGpL'] for i in xrange(len(b))],False)
if have_exp:
    ax[0].plot(H['GX'][:,0],H['GX'][:,1],'b.',ms=3)
    ax[1].plot(H['XG'][:,0],H['XG'][:,1],'b.',ms=3)
    ax[2].plot(H['GL'][:,0],H['GL'][:,1],'b.',ms=3)
```python
dos(ax[3], 2)
savefig('%s-disp.png'%pf)

b = ['L0', 'T1', 'T2']
fig = figure()
ax = subplots(fig)
curve(ax[0], [[b[i]+'pGpX', b[i]+'sGpX'] for i in xrange(len(b))], False)
curve(ax[1], [[b[i]+'sXpG', b[i]+'sXsG'] for i in xrange(len(b))], False)
curve(ax[2], [[b[i]+'pGpL', b[i]+'tGpL'] for i in xrange(len(b))], False)
if have_exp:
    ax[0].plot(H['GX'][:,0], H['GX'][:,1], 'b.', ms=3)
    ax[1].plot(H['XG'][:,0], H['XG'][:,1], 'b.', ms=3)
    ax[2].plot(H['GL'][:,0], H['GL'][:,1], 'b.', ms=3)
dos(ax[3], 1)
savefig('%s-disp+spq.png'%pf)

fig = figure(figsize=(4, 3))
DOS = N.variables['DOS'].data[:]
sDOS = smooth(DOS, 3)
sDOS /= sDOS.max()
plot(f, sDOS, 'r-', lw=2, alpha=0.75, label='DOS: \( f \)')
plot(2*f, sDOS, 'r:', lw=2, alpha=0.75, label=r'DOS: \( 2f \)')
SPK = N.variables['SPK'].data[:]
sSPK = smooth(SPK, 3)
sSPK /= sSPK.max()
plot(f, sSPK, 'b-', lw=2, alpha=0.75, label='SPQ')
xlim(0, f.max())
ylim(0, 1.4)
xlabel(r'Frequency \( f \) [THz]')
xticks([])
legend(loc='best', fancybox=True, mode='expand', ncol=3, handletextpad=0)
tight_layout()
savefig('%s-dos+spq.svg'%pf)
```

B.4 Thermal Expansion and Specific Heat Calculation

Listing B.10: Thermal expansion job script

```bash
#!/bin/bash

PBS -N EXPANSION
PBS -l procs=8
PBS -l walltime=01:00:00
PBS -q short
PBS -M horne.kyle@gmail.com
PBS -m a
PBS -t 1-1

# IIs="Al Ga In AlGa AlIn GaIn"
IIs="QCL"
R=8

for III in $IIs;
    do
        BASE_DIR=
BASE_DIR=$DORC_HOME/Expansion
        COMP_DIR=
COMP_DIR=$BASE_DIR/$III
        WORK_DIR=
WORK_DIR=$COMP_DIR/k=
WORK_DIR=$PBS_ARRAYID
        if [ ! -e $COMP_DIR ]; then
            mkdir $COMP_DIR
        fi
        if [ -e $WORK_DIR ]; then
            rm -rf $WORK_DIR
        fi
        mkdir $WORK_DIR && cd $WORK_DIR
        ../../generate.py
    # ../../ build.py -t_relax 10 -t_sim 10 -R0 $R -R1 $R -R2 $R -III $III -seed 123456
        ../../build.py -t_relax 10 -t_sim 10 -R0 $R -R1 $R -R2 $R -QCL -seed 123456
        mpirun -np $PBS_NP lammps -in ../../TEC.lmp -var k $PBS_ARRAYID
        done
```
Listing B.11: Thermal expansion lammps script

```plaintext
#====================================================#
# Input file for thermal expansion and specific heat #
#====================================================#

units metal
atom_style atomic
boundary p p p

include vars.inc

variable m equal mass(all)
read_data struct.xyz

group Al type 1
group Ga type 2
group In type 3
group As type 4

pair_style tersoff
pair_coeff * * ${coeffs} Al Ga In As

neighbor 2 bin
neigh_modify delay 5
velocity all create ${T0} ${seed} rot yes sum yes mom yes dist gaussian loop geom

timestep ${dt}

#===============================================#
# Setup log output #
#===============================================#

thermo $(steps_thermo)
thermo_style custom v_t temp etotal v_a

#===============================================#
# Setup variables and computes for output #
#===============================================#

variable a equal (lx/${R0}+ly/${R1}+lz/${R2})/3

variable kT00 equal ${T00}+400*((${k}-1)/10)-200
variable kPE equal pe
variable kKE equal ke
compute TEMP all temp
print "Set Temp = ${kT00}"

#===============================================#
# Relax the lattice to the potential in NTP #
#===============================================#

fix NPT all npt temp ${kT00} ${kT00} ${tau_T} iso ${P00} ${P00} ${tau_P}
reset_timestep 0
```
run $(\text{steps\_relax})$

# Simulate system and output average lattice parameter and energies #

fix AVE all ave/time $(\text{steps\_relax})$ $(\text{steps\_relax})$ &
c_TEMP v_a v_kPE v_kKE v_m file ave.dat
dump VELX_OUT all xyz $(\text{steps\_dump})$ $(\text{steps\_dump})$ ixv.$(\text{kT00}).xyz$

run $(\text{steps\_sim})$
Listing B.12: Thermal expansion post-processing script

```bash
#!/bin/bash
#
PBS -N EXPANSION
PBS -l procs=8
PBS -l walltime=01:00:00
PBS -q short
PBS -M horne.kyle@geatl.com
PBS -m a
PBS -t 1-1

IIIs="Al␣Ga␣In␣AlGa␣AlIn␣GaIn"

for III in \$
{IIIs};
do

BASE_DIR=$
{DORC_HOME}/Expansion
COMP_DIR=$
{BASE_DIR}/$ III

cd $
{COMP_DIR}
../post.py
mv combined.mat ../$ III.mat

done
```
Listing B.13: Thermal expansion plotting script

```python
#!/usr/bin/env python
import matplotlib as mpl
#~ mpl.rcParams['font.family'] = 'serif'
#~ mpl.rcParams['font.size'] = 13
#~ mpl.rcParams['font.serif'] = 'Times New Roman'
#~ mpl.rcParams['image.cmap'] = 'gray_r'
#~ mpl.rcParams['figure.figsize'] = '6,4'
mpl.rcParams['font.family'] = 'sans'
#~ mpl.rcParams['font.size'] = 13
mpl.rcParams['font.serif'] = 'Times New Roman'
mpl.rcParams['image.cmap'] = 'coolwarm'
mpl.rcParams['figure.figsize'] = '4,3'
from pylab import *
from scipy.io import loadmat
from scipy.optimize import curve_fit
from sys import argv
import os.path

if len(argv)<2:
    print 'Error: Too few arguments.'
    print 'Specify .mat file to read.'
    quit()

D = loadmat(argv[1])
pf = argv[1].split('.')[:0]
if os.path.isfile(ef):
    Te,Ae = loadtxt(ef,unpack=True)
else:
    Te = array([])
    Ae = array([])

if os.path.isfile(Cf):
    eCp = loadtxt(Cf)
else:
    eCp = -1.0

af = lambda T,c,l,q: c+l*T+q*T**2
da = lambda T,c,l,q: l+2*q*T
Af = lambda T,c,l,q: da(T,l,q)/af(T,c,l,q)

aC,aV = curve_fit(af,D['S'][:,0],D['S'][:,1])
eC,eV = curve_fit(af,D['S'][:,0],D['S'][:,2])

a = lambda T: af(T,aC[0],aC[1],aC[2])
e = lambda T: af(T,eC[0],eC[1],eC[2])
A = lambda T: 1.0E6*Af(T,aC[0],aC[1],aC[2])
C = lambda T: df(T,eC[1],eC[2])/D['m'][0,0]

N = 1000
T = linspace(D['S'][:,0].min(),D['S'][:,0].max(),N)
```

figure('Lattice Constant')
plot(T,a(T),'r-',lw=2,label='Fit')
plot(D['S'][0,0],D['S'][0,1],'bo',ms=2,label='Raw')
xlabel(r'Temperature $T$ [K]')
ylabel(r'Lattice Constant $a$ [Å]')
legend(loc='best',fancybox=True)
tight_layout()
savefig(pf+'-lattice.svg')

figure('Thermal Expansion')
plot(T,A(T),'r-',lw=2,label='Sim')
print '%6sAs $A = s: %.1f \times 10^{-6}$'%(pf,A(300))
if Te.size>0:
    plot(Te,1.0e6*Ae,'b.',label='Exp')
    ylim(min(yl[0],0),yl[1])
xlabel(r'Temperature $T$ [K]')
ylabel(r'Thermal Expansion $\alpha_L$ [K$^{-1} \times 10^{-6}$]')
legend(loc='best',fancybox=True)
tight_layout()
savefig(pf+'-expansion.svg')

figure('Total Energy')
plot(T,e(T),'r-',lw=2,label='Fit')
plot(D['S'][0,0],D['S'][0,2],'bo',ms=2,label='Raw')
xlabel(r'Temperature $T$ [K]')
ylabel(r'Total Energy $E$ [eV]')
legend(loc='best',fancybox=True)
tight_layout()
savefig(pf+'-energy.svg')

figure()
sCp = 96.485E6*C(300)
plot(T,96.485E6*C(T),'r-',lw=2,label='Simulation')
axhline(sCp,color='k',lw=2,ls='--',label='Sim @ 300 K')
print '%6sAs $C_p = (e: %.1f, s: %.1f) [+%.2f %%]'%(pf,eCp,sCp,(sCp-eCp)/eCp*100)
if eCp>0.0:
    axhline(eCp,color='b',lw=2,ls='--',label='Exp @ 300 K')
xlim(T.min(),T.max())
    ylim(0,1.05*yl[1])
xlabel(r'Temperature $T$ [K]')
ylabel(r'Specific Heat $C_p$ [J/kg·K]')
legend(loc='best',fancybox=True)
tight_layout()
savefig(pf+'-specific_heat.svg')

#show()
Appendix C

Dispersion Post Processing

C.1 Lammps Binary Reader

Listing C.1: Lammps binary reader and netCDF writer

```c
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "string.h"
#include "dump.h"
#include "netcdf.h"

int writeNC(const char* fi, const char* fo, int sort) {
    int ier;

    FILE* fi_h = fopen(fi,"rb");
    if(!fi_h) {printf("Error: Failed to open file \[\%s\]\n",fi); return 1;}

    snapshot_t* cur = readSnapshot(fi_h,sort);

    int fo_h;
    int dim_h[3];
    int dim2_h[3];
    int var_h,bounds_h,triclinic_h,step_h;

    ier = nc_create(fo,NC_CLOBBER,&fo_h);
    if(ier) {printf("Error: creating file\n"); return 1;}

    ier = nc_def_dim(fo_h,"col",cur->cols,&(dim_h[2]));
    ier = nc_def_dim(fo_h,"atom",cur->atomCount ,&(dim_h[1]));
    ier = nc_def_dim(fo_h,"time",NC_UNLIMITED ,&(dim_h[0]));

    ier = nc_def_dim(fo_h,"i",3,&(dim2_h[2]));
    ier = nc_def_dim(fo_h,"e",2,&(dim2_h[1]));
    dim2_h[0] = dim_h[0];

    int dim3_h[2] = {dim_h[0],dim2_h[2]};

    ier = nc_def_dim(fo_h,"step",NC_INT,1,dim_h,&step_h);
    ier = nc_def_dim(fo_h,"bounds",NC_DOUBLE,3,dim2_h,&bounds_h);
    ier = nc_def_dim(fo_h,"triclinic",NC_DOUBLE,2,dim3_h,&triclinic_h);
    ier = nc_def_dim(fo_h,"data",NC_FLOAT,3,dim_h,&var_h);

    ier = nc_def_var(fo_h,"step",NC_INT,1,dim_h,&step_h);
    ier = nc_def_var(fo_h,"bounds",NC_DOUBLE,3,dim2_h,&bounds_h);
    ier = nc_def_var(fo_h,"triclinic",NC_DOUBLE,2,dim3_h,&triclinic_h);
    ier = nc_def_var(fo_h,"data",NC_FLOAT,3,dim_h,&var_h);

    ier = nc_enddef(fo_h);
}
```
size_t start[3] = {0,0,0};
size_t count[3] = {1,(size_t) cur->atomCount,(size_t) cur->cols};
size_t count2[3] = {1,2,3};

size_t tstart[2] = {0,0};
size_t tcount[2] = {1,3};

while(cur) {
    printf("Reading step: %zd\n",start[0]);
    int step = cur->timeStep;
    ier = nc_put_vara_int(fo_h,step_h,start,count,&step);
    double bnd[2][3];
    double tri[3] = {cur->xy,cur->xz,cur->yz};
    int ie;
    for(ie=0;ie<2;ie++) {
        bnd[ie][0] = cur->xb[ie];
        bnd[ie][1] = cur->yb[ie];
        bnd[ie][2] = cur->zb[ie];
    }
    ier = nc_put_vara_double(fo_h,bounds_h,start,count2,&(bnd[0][0]));
    ier = nc_put_vara_double(fo_h,triclinic_h,tstart,tcount,&(tri[0]));
    ier = nc_put_vara_double(fo_h,var_h,start,count,cur->data);
    freeSnap(cur);
    cur = readSnapshot(fi_h,sort);
    start[0]++;
}

nc_close(fo_h);
fclose(fi_h);
return 0;
}

int main(int argc, char* argv[]) {
    char* fn;
    int sort = FALSE;
    int ier;

    switch(argc) {
    case 0:
    case 1:
        printf("Error: Too few arguments\n");
        return 1;
    break;
    case 2:
        fn = argv[1];
        break;
    case 3:
        if(strcmp(argv[1],"-s")==0) sort = TRUE;
        else printf("Warning: Unrecognized argument [%s]\n",argv[1]);
        fn = argv[2];
        break;
    default:

break;
}

ier = writeNC(fn,"lamps.nc",sort);
if(ier) {printf("Error writing file\n"); return 1;}

return 0;
C.2 Density of States Calculation

Listing C.2: Density of states calculation

```c
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "complex.h"
#include "fftw3.h"
#include "dump.h"
#include "netcdf.h"
#include "omp.h"

#define PI 3.1415359

int writeDOS(int fi_h, int fo_h) {
    int ier;

    size_t timeSteps, atomCount, freqCount;
    int dim_time_h, dim_atom_h, dim_freq_h;
    int data_h, var_h;

    ier = nc_inq_dimid(fi_h, "time", &dim_time_h);
    ier = nc_inq_dimlen(fi_h, dim_time_h, &timeSteps);
    ier = nc_inq_dimid(fi_h, "atom", &dim_atom_h);
    ier = nc_inq_dimlen(fi_h, dim_atom_h, &atomCount);
    ier = nc_inq_varid(fi_h, "data", &data_h);

    ier = nc_redef(fo_h);
    int dim_h[1] = {dim_freq_h};
    ier = nc_def_var(fo_h, "DOS", NC_DOUBLE, 1, dim_h, &var_h);
    ier = nc_enddef(fo_h);

    int iw;

    double* D = (double*) malloc(freqCount*sizeof(double));
    for(iw=0; iw<freqCount; iw++) D[iw] = 0.0;

    #pragma omp parallel default(shared) private(iw, ier)
    {
        int ia, id, it, ic;
        double* uvw;
        double* in;
        complex* out;
        fftw_plan p;
        int tid = 0;
        int Nt = 1;

        #pragma omp
        {
            tid = omp_get_thread_num();
        }
    }
```

Nt = omp_get_num_threads();

double* Dl = (double*) malloc(freqCount*sizeof(double));
for(iw=0;iw<freqCount;iw++) Dl[iw] = 0.0;

uvw = (double*) malloc(timeSteps*3*sizeof(double));
#pragma omp critical
{
    in = (double*) fftw_malloc(timeSteps*sizeof(double));
    out = (complex*) fftw_malloc(freqCount*sizeof(complex));
}

#pragma omp critical
(p = fftw_plan_dft_r2c_1d(timeSteps,in,out,FFTW_MEASURE);)

#pragma omp critical
for(ia=0;ia<atomCount;ia++) {
   if(tid==0) {
        printf("\[␣␣␣DOS\]␣|"');
        for(ic=0;ic<50;ic++) printf(ic*atomCount<ia*50?"=":"␣"');
        printf("|␣%.0f␣%%",100*((double) ia)/((double) atomCount));
   }
}

size_t start[3] = {0,ia,6};
size_t count[3] = {timeSteps,1,3};
#pragma omp critical
(ier = nc_get_vara_double(fi_h,data_h,start,count,uvw);)

#pragma omp critical
for(it=0;it<timeSteps;it++) {
    for(it=0;it<timeSteps;it++) in[it] = uW[3*it+id];
    fftw_execute(p);
    for(iw=0;iw<freqCount;iw++) Dl[iw] += pow(cabs(out[iw]),2);
}
#pragma omp critical
(for(iw=0;iw<freqCount;iw++) D[w] += Dl[iw];)

ier = nc_put_vara_double(fo_h,var_h,&D[w];)
printf("\n")
return ier;
C.3 Spectral Conductivity Calculation

Listing C.3: Spectral conductivity calculation

```c
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "complex.h"
#include "fftw3.h"
#include "dump.h"
#include "netcdf.h"
#include "omp.h"

#define PI 3.14159

int writeSPK(int fi_h, int fo_h) {
    int ier;

    size_t timeSteps, atomCount, freqCount;
    int dim_time_h, dim_atom_h, dim_freq_h;
    int data_h, var_h;

    ier = nc_inq_dimid(fi_h, "time", &dim_time_h);
    ier = nc_inq_dimlen(fi_h, dim_time_h, &timeSteps);
    ier = nc_inq_dimid(fi_h, "atom", &dim_atom_h);
    ier = nc_inq_dimlen(fi_h, dim_atom_h, &atomCount);

    ier = nc_redef(fo_h);
    int dim_h[1] = {dim_freq_h};
    ier = nc_def_var(fo_h, "SPK", NC_DOUBLE, 1, dim_h, &var_h);
    ier = nc_enddef(fo_h);

    int iw;

    double* D = (double*) malloc(freqCount*sizeof(double));
    for(iw=0; iw<freqCount; iw++) D[iw] = 0.0;

    #pragma omp parallel default(shared) private(iw, ier)
    {
        int ia, id, it, ic;
        double* qxyz;
        double* in;
        complex* out;
        fftw_plan p;
        int tid = 0;
        int Nt = 1;

        #pragma omp
        {
            tid = omp_get_thread_num();
```

Nt = omp_get_num_threads();

double* Dl = (double*) malloc(freqCount*sizeof(double));
for(iw=0;iw<freqCount;iw++) Dl[iw] = 0.0;
qxyz = (double*) malloc(timeSteps*3*sizeof(double));
#pragma omp critical
{
in = (double*) fftw_malloc(timeSteps*sizeof(double));
out = (complex*) fftw_malloc(freqCount*sizeof(complex));
}
#pragma omp critical
{
p = fftw_plan_dft_r2c_1d(timeSteps,in,out,FFTW_MEASURE);
}
#pragma omp for schedule(static,1)
for(ia=0;ia<atomCount;ia++) {
    if(tid==0) {
        printf("\[␣␣␣SPK\]␣|");
        for(ic=0;ic<50;ic++) printf(ic*atomCount<ia*50?"=":"␣");
        printf("|␣%.0f␣%%",100*((double) ia)/((double) atomCount));
    }
    size_t start[3] = {0,ia,9};
    size_t count[3] = {timeSteps ,1,3};
    #pragma omp critical
    {ier = nc_get_vara_double(fi_h,data_h,start,count,qxyz);}
    for(it=0;it<timeSteps;it++) { for(id=0;id<3;id++) {
        for(iw=0;iw<freqCount;iw++) Dl[iw] += pow(cabs(out[iw]),2);
    }
    #pragma omp critical
    {for(iw=0;iw<freqCount;iw++) D[iw] += Dl[iw];
    }
    ier = nc_put_vara_double(fo_h,var_h,&(D[0]));
    printf("\n");
    return ier;
}
C.4 Dispersion Calculation

Listing C.4: Dispersion calculation

```c
#include "disp.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"
#include "complex.h"
#include "fftw3.h"
#include "dump.h"
#include "netcdf.h"

int path(int fi_h, int fo_h, const char* an, int N, double k1[], double k2[], double n1[], double n2[], double n3[]) {
    char buf[7] = "B#K1K2";
    int i,ier;
    for(k=0;k<4;k++) buf[k+2] = an[k];
    buf[0] = 'L'; buf[1] = '0';
    ier = writeDSP(fi_h,fo_h,buf,N,k1,k2,n1);
    buf[0] = 'T'; buf[1] = '1';
    ier = writeDSP(fi_h,fo_h,buf,N,k1,k2,n2);
    buf[0] = 'T'; buf[1] = '2';
    ier = writeDSP(fi_h,fo_h,buf,N,k1,k2,n3);
    return ier;
}

int writeDSP(int fi_h, int fo_h, const char* an, int N, double k1[], double k2[], double n[]) {
    int ier;
    size_t timeSteps,atomCount;
    int dim_time_h,dim_atom_h;
    int dim_nu_h,dim_k_h;
    int data_h,var_h;
    ier = nc_inq_dimid(fi_h,"time",&dim_time_h);
    ier = nc_def_dim(fo_h,an,N,&dim_k_h);
    ier = nc_def_var(fo_h,an,NC_DOUBLE,2,dim_h,&var_h);
ieder = nc_enddef(fo_h);
```
int ik, it, ia, ix, ic;
complex ins[timeSteps][N];
double outs[timeSteps][N];

double* data = (double*) malloc(atomCount*6*sizeof(double));
// double* data = (double*) malloc(atomCount*9*sizeof(double));
#pragma omp parallel default(shared) private(ik, it, ia, ix, ic)
{
    for(it=0; it<timeSteps; it++) {
        //pragma omp master
        {
            printf("\r[%s] \n", an);
            for(ic=0; ic<50; ic++) printf(ic*timeSteps<it*50?"=":" ");
            printf("| \n\n %0.0f \n%%", 100*((double) it)/((double) timeSteps));
            fflush(stdout);
            size_t start[3] = {it, 0, 3};
            size_t count[3] = {1, atomCount, 6};
            size_t count[3] = {1, atomCount, 9};
            ier = nc_get_vara_double(fi_h, data_h, start, count, data);
        }
        #pragma omp barrier
        #pragma omp for
        for(ik=0; ik<N; ik++) {
            double k[3];
            for(ix=0; ix<3; ix++)
                k[ix] = (k2[ix]-k1[ix])*(double) ik / (double) (N-1)+k1[ix];
            ins[it][ik] = 0.0;
            for(ia=0; ia<atomCount; ia++) {
                double vn = 0.0;
                for(ix=0; ix<3; ix++)
                    vn += data[6*ia+ix+3]*n[ix];
                double kx = 0.0;
                for(ix=0; ix<3; ix++)
                    kx += k[ix]*data[6*ia+ix];
                ins[it][ik] += vn*cexp(-I*kx);
                // ins[it][ik] += cexp(-I*kx); // Compute Structure Factor -- Less Effective!
            }
        }
        free(data);
        // for(it=0; it<timeSteps; it++) for(ik=0; ik<N; ik++)
        // printf("%f\n", cabs(ins[it][ik]));
        complex* in = (complex*) fftw_malloc(timeSteps*sizeof(complex));
        complex* out = (complex*) fftw_malloc(timeSteps*sizeof(complex));
        fftw_plan plan = fftw_plan_dft_1d(timeSteps, in, out, FFTW_FORWARD, FFTW_MEASURE);
        for(ik=0; ik<N; ik++) {
            for(it=0; it<timeSteps; it++)
                in[it] = ins[it][ik];
            outs[it][ik] = 0.0;
            fftw_execute(plan);
            for(it=0; it<timeSteps/2; it++)
                outs[it][ik] = cabs(out[it]);
        }
        ier = nc_put_vara_double(fo_h, var_h, &outs[0][0]);
        printf("\n");
        return ier;
    }
}
void norm(double p0[], double p1[], double n[]) {
    int k;
    double m;
    for(k=0;k<3;k++) n[k] = p1[k]-p0[k];
    m = sqrt(n[0]*n[0]+n[1]*n[1]+n[2]*n[2]);
    for(k=0;k<3;k++) n[k] /= m;
}
C.5 Driver for FCC Systems

Listing C.5: FCC system driver

```c
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"
#include "complex.h"
#include "fftw3.h"
#include "netcdf.h"
#include "disp.h"
#include "dos.h"

void compute_N(double Nx, double Ny, double Nz, int* N);
void get_args(int argc, char* argv[], char* fn, int* Nx, double* dt);
void prepare_files(const char* fn, int Nx, double dt, int* fi_h, int* fo_h, double* a, int* Ny, int* Nz);
void finish_files(int* fi_h, int* fo_h);

int main(int argc, char* argv[]) {
    int ier;
    char fn[1024]; int Nx,Ny,Nz; double dt;
    get_args(argc,argv,fn,&Nx,&dt);

    int fi_h,fo_h; double a;
    prepare_files(fn,Nx,dt,&fi_h,&fo_h,&a,&Ny,&Nz);

    ier = writeDOS(fi_h,fo_h);
    ier = writeSPK(fi_h,fo_h);

    int N[3];
    compute_N(Nx,Nx,Nx,N);

    int pN = N[0];
    int sN = N[1];
    int tN = N[2];

    double n[3];
    double pG[3] = {0.0,0.0,0.0};
    double sG[3] = {4.0*PI/a,0.0,0.0};
    double tG[3] = {2.0*PI/a,2.0*PI/a,2.0*PI/a};
    double pX[3] = {2.0*PI/a,0.0,0.0};
    double sX[3] = {2.0*PI/a,2.0*PI/a,0.0};
    double pL[3] = { PI/a, PI/a, PI/a};

    norm(pG,pX,n);
    ier = writeDSP(fi_h,fo_h, "L0pGpX", pG,pG,pX,n);
    norm(sX,pX,n);
    ier = writeDSP(fi_h,fo_h, "T1pGpX", pG,pG,pX,n);
    norm(sX,tG,n);
    ier = writeDSP(fi_h,fo_h, "T2pGpX", pG,pG,pX,n);
    norm(sG,pX,n);
    ier = writeDSP(fi_h,fo_h, "L0sGpX", pG,sG,pX,n);
    norm(pX,sX,n);
```
ier = writeDSP(fi_h,fo_h,"T1sGpX",pX,sG,pX,n);
54 norm(sX,tG,n);
ier = writeDSP(fi_h,fo_h,"T2sGpX",pX,sG,pX,n);
56 norm(sX,pG,n);
ier = writeDSP(fi_h,fo_h,"L0sXpG",sX,sX,pG,n);
58 norm(sG,ax,n);
ier = writeDSP(fi_h,fo_h,"T1sXpG",sX,sX,pG,n);
60 norm(ax,tG,n);
ier = writeDSP(fi_h,fo_h,"T2sXpG",sX,sX,pG,n);
62 norm(sX,pG,n);
ier = writeDSP(fi_h,fo_h,"L0sXsG",sX,sX,sG,n);
64 norm(sG,sX,n);
ier = writeDSP(fi_h,fo_h,"T1sXsG",sX,sX,sG,n);
66 norm(sX,tG,n);
ier = writeDSP(fi_h,fo_h,"T2sXsG",sX,sX,sG,n);
68 norm(sX,sG,n);
ier = writeDSP(fi_h,fo_h,"L0sXpG",sX,sX,pG,n);
70 norm(pG,pL,n);
ier = writeDSP(fi_h,fo_h,"L0pGl",tG,pG,pL,n);
72 norm(ax,ax,n);
ier = writeDSP(fi_h,fo_h,"T1pGl",tG,pG,pL,n);
74 norm(ax,pl,n);
ier = writeDSP(fi_h,fo_h,"T2pGl",tG,pG,pL,n);
76 norm(tG,pL,n);
ier = writeDSP(fi_h,fo_h,"L0tGl",tG,tG,pL,n);
78 norm(sG,ax,n);
ier = writeDSP(fi_h,fo_h,"T1tGl",tG,tG,pL,n);
80 norm(pL,ax,n);
ier = writeDSP(fi_h,fo_h,"T2tGl",tG,tG,pL,n);
82 norm(tG,pL,n);
finish_files(&fi_h,&fo_h);
85 return ier;
}

void compute_N(double Nx, double Ny, double Nz, int* N) {
// Finally Correct for cubic (Nx=Ny=Nz)
N[0] = Nx+1;
N[1] = Nx+1;
N[2] = Nx/2+1;
printf("%d␣%d␣%d
",N[0],N[1],N[2]);
}

void get_args(int argc, char* argv[], char* fn, int* Nx, double* dt) {
if(argc<4) {
printf("Error:␣Too␣few␣arguments
");
printf("disp␣{dump.nc}␣{Nx}␣{dt}
");
exit(1);
}
scanf(argv[1],"%s",fn);
scanf(argv[2],"%d",Nx);
scanf(argv[3],"%lf",dt);
void prepare_files(const char* fn, int Nx, double dt, int* fi_h, int* fo_h, double* a, int* Ny, int* Nz) {
    int ier;
    int dim_time_h;
    ier = nc_open(fn, NC_NOWRITE, fi_h);
    int dim_time_h;
    size_t timeSteps;
    ier = nc_inq_dimid(*fi_h, "time", &dim_time_h);
    size_t timeSteps;
    ier = nc_inq_dimlen(*fi_h, dim_time_h, &timeSteps);
    int var_bnd_h;
    ier = nc_inq_varid(*fi_h, "bounds", &var_bnd_h);
    int var_step_h;
    ier = nc_inq_varid(*fi_h, "step", &var_step_h);
    int* steps = (int*) malloc(timeSteps*sizeof(int));
    ier = nc_get_var_int(*fi_h, var_step_h, steps);
    double SR;
    SR = 1.0/(1.0E-12*dt*(steps[1]-steps[0]));
    free(steps);
    double bnd[2][3];
    size_t start[3] = {0,0,0};
    size_t count[3] = {1,2,3};
    ier = nc_get_vara_double(*fi_h, var_bnd_h, start, count, &bnd[0][0]);
    *a = (bnd[1][0]-bnd[0][0])/Nx;
    *Ny = (bnd[1][1]-bnd[0][1])/*a;*/
    *Nz = (bnd[1][2]-bnd[0][2])/*a;*/
    ier = nc_create("L.nc", NC_CLOBBER, fo_h);
    int dim_nu_h;
    ier = nc_def_dim(*fo_h, "nu", timeSteps/2+1, &dim_nu_h);
    int var_f_h;
    ier = nc_def_var(*fo_h, "f", NC_DOUBLE, 1, &dim_nu_h, &var_f_h);
    ier = nc_enddef(*fo_h);
    double* freq = (double*) malloc((timeSteps/2+1)*sizeof(double));
    int k;
    for(k=0;k<timeSteps/2+1;k++) freq[k] = ((double) k)/((double) (timeSteps/2))*SR/2.0;
    ier = nc_put_var_double(*fo_h, var_f_h, freq);
    free(freq);
    if(ier) printf("Error\n");
}

void finish_files(int* fi_h, int* fo_h) {
    nc_close(*fo_h);
    nc_close(*fi_h);
}
C.6   Driver for QCL Systems

Listing C.6: QCL system driver

```c
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"
#include "complex.h"
#include "fftw3.h"
#include "netcdf.h"
#include "disp.h"
#include "dos.h"

void compute_N(double Nx, double Ny, double Nz, int* N);
void get_args(int argc, char* argv[], char* fn, int* Nx, double* dt);
void prepare_files(const char* fn, int Nx, double dt, int* fi_h, int* fo_h, double* a, int* Ny, int* Nz);
void finish_files(int* fi_h, int* fo_h);

int main(int argc, char* argv[]) {
  int ier;
  char fn[1024]; int Nx,Ny,Nz; double dt;
  get_args(argc,argv,fn,&Nx,&dt);
  
  int fi_h,fo_h; double a;
  prepare_files(fn,Nx,dt,&fi_h,&fo_h,&a,&Ny,&Nz);
  
  ier = writeDOS(fi_h,fo_h);
  ier = writeSPK(fi_h,fo_h);

  double n1[3],n2[3],n3[3];
  double s = 2.0*PI/a;
  
  double Go[3] = {0.0*s,0.0*s,0.0*s};
  double GC[3] = {2.0*s,0.0*s,0.0*s};
  double GP[3] = {0.0*s,2.0*s,0.0*s};
  double GL[3] = {1.0*s,1.0*s,1.0*s};
  
  double Xc[3] = {1.0*s,0.0*s,0.0*s};
  double XC[3] = {1.0*s,0.0*s,1.0*s};
  double Xp[3] = {0.0*s,1.0*s,0.0*s};
  double XP[3] = {0.0*s,1.0*s,1.0*s};
  
  double Lo[3] = {0.5*s,0.5*s,0.5*s};
  double Ko[3] = {0.75*s,0.75*s,0.0*s};

  int N[4];
  compute_N((double) Nx,(double) Ny,(double) Nz,N);

  // Delta_c
  norm(Go,Xc,n1); norm(Xp,Go,n2); norm(Xc,XC,n3);
  path(fi_h,fo_h,"GoXc",N[0],Go,Xc,n1,n2,n3);
  norm(GC,Xc,n1); norm(Go,Xp,n2); norm(Xc,XC,n3);
  path(fi_h,fo_h,"GCXc",N[0],GC,Xc,n1,n2,n3);
```

// Sigma_c
norm(XC, Go, n1); norm(Go, Xp, n2); norm(GC, XC, n3);
path(fi_h, fo_h, "XCGo", N[1], XC, Go, n1, n2, n3);

// Delta_p
norm(Go, Xp, n1); norm(Go, Xc, n2); norm(Xc, XC, n3);
path(fi_h, fo_h, "GoXp", N[1], Go, Xp, n1, n2, n3);

// Sigma_p
norm(XP, Go, n1); norm(Xc, Go, n2); norm(GP, XP, n3);
path(fi_h, fo_h, "XPGo", N[1], XP, Go, n1, n2, n3);

// Lambda
norm(Go, Lo, n1); norm(Xp, Xc, n2); norm(Ko, Lo, n3);
path(fi_h, fo_h, "GoLo", N[2], Go, Lo, n1, n2, n3);

finish_files(&fi_h, &fo_h);

return ier;

}

void compute_N(double Nx, double Ny, double Nz, int* N) {
    N[0] = (Nx+1);
    N[1] = (Ny+1);
    N[2] = (Ny/2+1);
    printf("%d %d %d\n", N[0], N[1], N[2]);
}

void get_args(int argc, char* argv[], char* fn, int* Nx, double* dt) {
    if(argc<4) {
        printf("Error: Too few arguments\n");
        exit(1);
    }
    sscanf(argv[1], "%s", fn);
    sscanf(argv[2], "%d", Nx);
    sscanf(argv[3], "%lf", dt);
}

void prepare_files(const char* fn, int Nx, double dt, int* fi_h, int* fo_h, double* a, int* Ny, int* Nz) {
    int ier;
    ier = nc_open(fn, NC_NOWRITE, fi_h);
    int dim_time_h;
ier = nc_inq_dimid(*fi_h,"time",&dim_time_h);
size_t timeSteps;
ier = nc_inq_dimlen(*fi_h,dim_time_h,&timeSteps);
int var_bnd_h;
ier = nc_inq_varid(*fi_h,"bounds",&var_bnd_h);
int var_step_h;
ier = nc_inq_varid(*fi_h,"step",&var_step_h);
int* steps = (int*) malloc(timeSteps*sizeof(int));
ier = nc_get_var_int(*fi_h,var_step_h,steps);
double SR;
SR = 1.0/(1.0E-12*dt*(steps[1]-steps[0]));
free(steps);

double bnd[2][3];
size_t start[3] = {0,0,0};
size_t count[3] = {1,2,3};
ier = nc_get_vara_double(*fi_h,var_bnd_h,start,count,&(bnd[0][0]));
*a = (bnd[1][0]-bnd[0][0])/(*a);
*Sy = (bnd[1][1]-bnd[0][1])/*a;*
*Sz = (bnd[1][2]-bnd[0][2])/*a;*
ier = nc_create("L.nc",NC_CLOBBER,fo_h);

int dim_nu_h;
ier = nc_def_dim(*fo_h,"nu",timeSteps/2+1,&dim_nu_h);
int var_f_h;
ier = nc_def_var(*fo_h,"f",NC_DOUBLE,1,&dim_nu_h,&var_f_h);
ier = nc_enddef(*fo_h);

double* freq = (double*) malloc((timeSteps/2+1)*sizeof(double));
int k;
for(k=0;k<timeSteps/2+1;k++) freq[k] = ((double) k)/((double) (timeSteps/2))*SR/2.0;
ier = nc_put_var_double(*fo_h,var_f_h,freq);
free(freq);
if(ier) printf("Error\n");
}

void finish_files(int* fi_h, int* fo_h) {
nc_close(*fo_h);
nc_close(*fi_h);
}
Vita

Kyle S. Horne

Published Journal Articles


Conference Talks and Posters


