Simulation of Quantum Cascade Lasers

Zakaria Mohamed

Thesis submitted for the degree of Bachelor of Science
Project duration: 2 months
May 2023

Supervised by Andreas Wacker

Mathematical Physics
Department of Physics

LUND UNIVERSITY

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Andreas Wacker, for his exceptional support and guidance throughout the entire duration of my thesis. His open door policy and insightful conversations played a crucial role in shaping my understanding and cultivating my interest in the subject, which makes me very lucky to be under his mentorship. I would also like to extend my heartfelt appreciation to my parents for their unwavering support and encouragement throughout my life, which has been invaluable to me, and for which i am fortunate to have. Additionally, I would like to acknowledge my colleague, Francessco Diotallevi, for his technical assistance and motivation during this thesis. His expertise, willingness to help, were instrumental in overcoming challenges and pushing me to work harder. I am grateful for his collaboration and friendship.

Abstract

This thesis aimed to contribute to the improvement of Quantum Cascade Lasers (QCLs) by focusing on the effective determination of quantum levels in these devices. The current method relies on an outdated Fortran code, which poses challenges when integrating into optimization schemes. To address this issue, the thesis proposed the development of an improved version of the code using Python, thus enhancing readability and flexibility.

Contents

List of Abbreviations

ULS	Upper Laser State
LLS	Lower Laser State
LO	Longitudinal Optical Phonon Energy
RT	Resonant Tunneling

1 Introduction

In the early 1970s, the discovery of the quantum well by Esaki and Tsu [5] paved the way for researchers to explore the unique properties that heterostructures exhibit and their potential use in existing technologies. One area of interest was the use of these structures for light amplification, which led to the development of the first Quantum Well Laser by Raymond Dingle at Bell Laboratories. Dingle suggested that the use of quantum well structures with layers of different band gaps allowed for a higher level of wavelength tunability, simply by altering the thickness of the layers , whereas traditional lasers required a change in layer composition, see Figure 1. This lead to the invention of the first Quantum Well Laser [4], however, the technology was still in its early stages, and the first Quantum Well Lasers suffered from several limitations, which further motivated researchers to explore new approaches that may overcome these challenges.

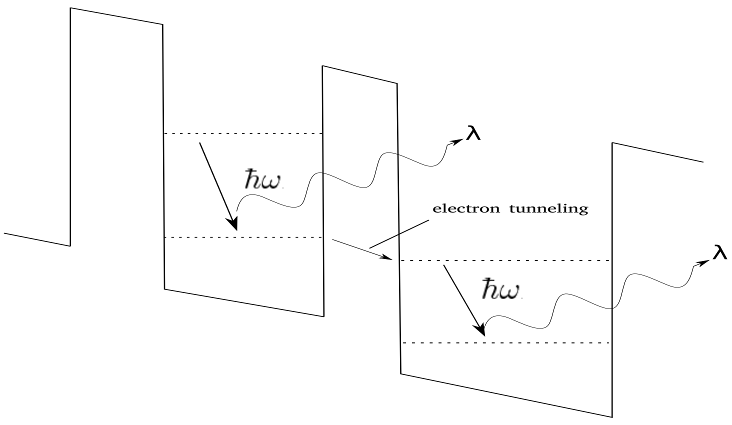

Figure 1: Simple diagram portraying the idea of using a heterostructure under bias for light amplification.

Through advances in band structure engineering, groundbreaking development in semiconductor laser technology was achieved with the invention of the Quantum Cascade Laser (QCL) by researcher Federico Capasso at Bell Laboratories [6]. Unlike conventional lasers that rely on interband transitions, QCLs use optical transitions between subbands within the semiconductor material, providing several key advantages. One such advantage is the emitted radiation being dependent on the structure's dimensions, with the subband gap determined by the size of the barrier. These subband gaps are also much smaller than the band gaps in materials, allowing QCLs to operate at lower frequencies. Initial designs emitted radiation within the infrared (IR) region of the optical spectrum, ranging from 3.4 to 24 μm (88 to 12.5 THz). However, these designs came with limitations such as the

Reststrahlen band, which sets a lower limit for emitted frequencies, additionally operations at room temperature were not yet possible [7].
Further breakthrough was realised in 2002, researchers accomplished continuous wave (cw) operation by IR-QCLs at room temperature, which was made possible by the two-phonon resonance design [1]. More significant was the development of THz-QCLs that operate below the Reststrahlen band, which prove to have the potential to unlock a host of new technological applications[13], albeit achieving room temperature THz-QCLs remains a current area of research.

2 Operation Principles of QCLs

The basic building blocks of a QCL are a series of quantum wells, designed by layering thin semiconductor material ($\sim$ 10Å order of magnitude) with varying bandgap energies. A very commonly employed material combination is GaAs as the well material and Al$_x$Ga$_{1-x}$As as the barrier material, where x represents the relative proportion of aluminum to gallium. At the heterojunctions of the structure, the conduction and valence band edges will bend, due to the discontinuity of the bands at the edges, forming the band offsets. These band offsets are the barriers of the Quantum Well, see Figure 2.

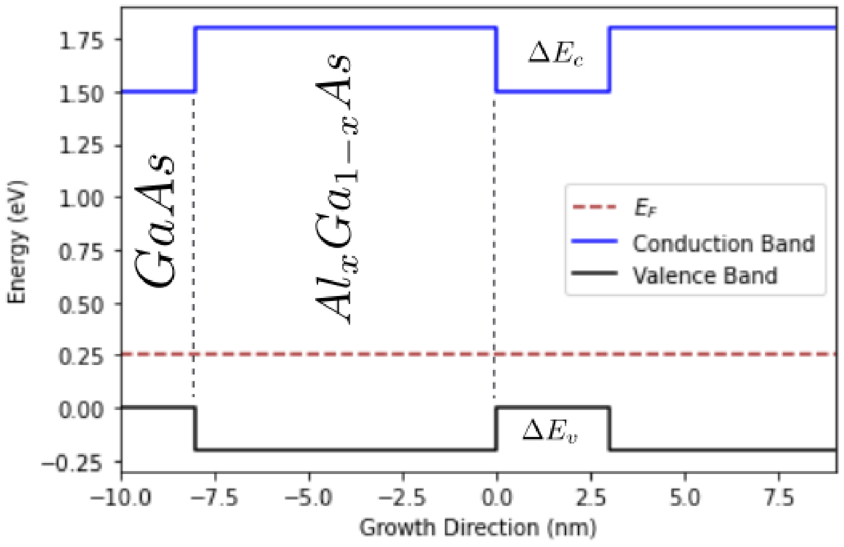

Figure 2: Schematic diagram of the offsets in a heterostructure. Here the conduction band offset acts as the barrier height.

The electron energy states now depend on the barrier height E_c and the thickness of the layers. These parameters may now be tuned to obtain the desired eigenstates.

The active region of a QCL, where stimulated emission takes place, consists of a minimum of 3 bands. These are commonly referred to as the Upper and lower laser levels, where light is emission takes place, and a 3rd subband known as the extractor level. The extractor level ensures that higher energy levels, in this case the Upper laser level, are re-populated after de-excitations to allow for further emissions to take place.

2.1 Electron Extraction Methods

The most prevalent and effective method currently utilized for depopulating the LLS is known as the direct-phonon method [12]. This method entails positioning the extractor level directly below the LLS, with a separation equivalent to one Longitudinal Optical (LO) phonon energy. By strategically positioning the extractor level below the LLS, transitions from the ULS to the extractor level happen at a much slower rate as the spatial overlap is much lower, resulting in significantly shorter lifetimes for the LLS state in comparison to the ULS. The LO-phonon scattering process is fast, causing subband lifetimes to be ~ 1 ps whenever LO-phonon scattering is energetically feasible. This helps maintain what is known as *population inversion*, which is necessary for continued stimulated emission. Different QCls designs are centered around his important concept, with the mid-IR and THz designs operating at freqeuncies above and below the longitudinal optical phonon energy E_{LO}.

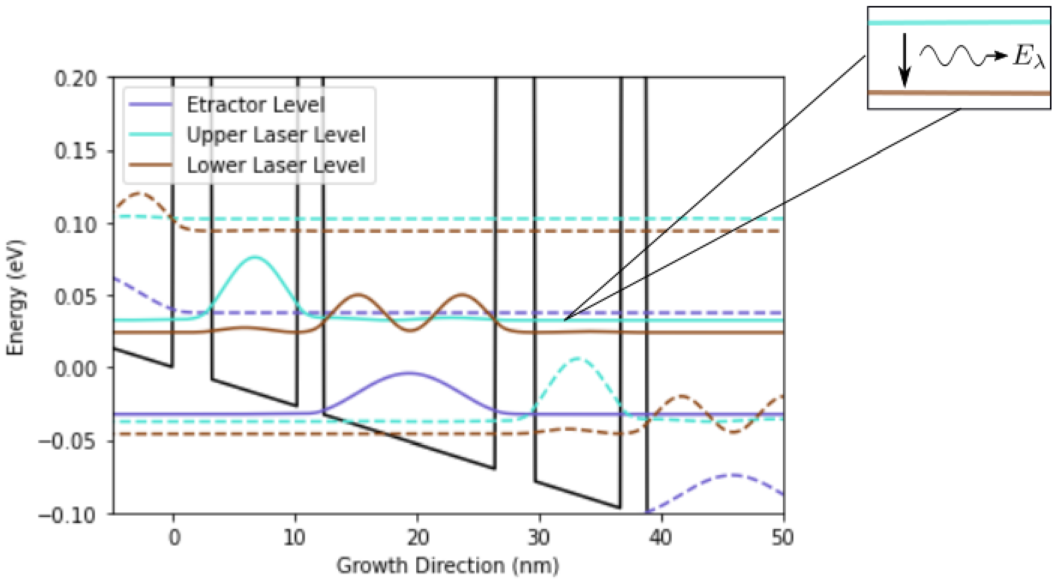

Figure 3: Diagram of the energy states in a THz QCL, here $E_\lambda < E_{LO}$ where E_{LO} is the energy between the LLS and the extractor level.

2.2 Electron Injection Methods

Electrons are then injected back into the ULS through Resonant Tunneling (RT). Resonant tunneling occurs when the energy levels of the adjacent quantum wells are aligned in such a way that electrons can tunnel through the barrier separating them with high probability. By carefully designing the QCL structure, the energy levels of the extractor state and the ULS are chosen to align under a bias, as illustrated in Figure 3. The width of the barrier plays a critical role in determining the rate at which electrons transfer through resonant tunneling. Specifically, the narrower the barrier width, the higher the rate of electron transition. However reducing the barrier width comes with its own drawbacks, in particular for low frequency devices, i.e. when the ULS and the LLS are in close proximity, there is a possibility for electrons to tunnel into the LLS instead of the ULS resulting in a higher current without the desired emission of a photon [8]. In such case Scattering-Assisted Injection (AS) has been shown to perform better [10], where electron injection into the ULS is also performed by the direct phonon method. In such case electrons are extracted from the LLS through resonant phonon extraction [18], and subsequently injected into the ULS, with the injector state 1 $\varDelta E_{LO}$ above the ULS.

3 Basis States

The electronic energy states in the periodic potential of a crystal structure is described by the Bloch theorem. It states that the wavefunction of an electron in a crystal can be expressed as a product of a plane wave and a periodic function that has the same periodicity as the crystal lattice [2]. The electron wavefunction can then be written as

$$\psi_{\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}} u_{\mathbf{k}}(\mathbf{r}) \tag{1}$$

Here, **k** is the wavevector, also known as the Bloch vector, taking values that lie within the Brillouin zone, and $u_{\mathbf{k}}(\mathbf{r})$ is a periodic function with the periodicity of the crystal lattice, satisfying the condition:

$$u_{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = u_{\mathbf{k}}(\mathbf{r}) \tag{2}$$

where **R** is any lattice vector.
By substituting the wavefunction expression from Eq. 1 into the Schrödinger equation,

$$\hat{H}\psi(\mathbf{r}) = E\psi(\mathbf{r}) \tag{3}$$

one can obtain the energy eigenvalues and eigenstates of the electrons in the crystal. The resulting energy bands depict the allowed energy levels for electrons as a function of $\mathbf{k}$.
While this theorem provides a powerful framework for understanding the electronic properties of crystalline materials, they may not be the most intuitive basis for understanding localized electronic states. This is due to the fact that these Bloch functions have an indeterminacy regarding their phase, meaning that they are oscillating and delocalized in real space. One may negate this delocalization by simply performing an arbitrary unitary transformation, subsequently arriving at a set of solutions that are equally valid.
In order to obtain useful microscopic insights into the chemical and physical processes of a structure, it is important to consider basis states that are localized in real space.

This localisation is achieved by performing a Fourier transform on the Bloch states, and inserting a new k-dependent phase factor, such that it is real for a given vector φ.

$$\omega(\mathbf{r}) = \frac{1}{N}\sum_{j} e^{i\mathbf{k}_j\cdot\varphi} u_{\mathbf{k}_j}(\mathbf{r}) \tag{4}$$

These states, described in Eq. 4, are known as the Wannier states [16]. Note that the Wannier states are not eigenstates of the Hamiltonian and as such trade localization in energy for localization in space, instead they are labelled by the unit cell they are localised too, and their band index ν.

3.1 Two Band Kane Model

Theoretical determination of electronic energy states has traditionally relied on solving for the eigenstates of the one-band envelope-function Hamiltonian, which focuses solely on the conduction band. This approach is accurate for energy levels located near the bottom of the conduction band. However, as the barrier energy becomes comparable to the energy gap between different bands, it becomes necessary to consider the contribution of the valence band due to the non-parabolicity of the conduction band.
To address this, a simplified two-band model, known as the Kane Model [15], has been introduced. Within this model, electronic energies can be calculated with remarkable agreement to experimental values by also incorporating an energy-dependent effective mass. The following part of this chapter essentially follows internal notes by Andreas Wacker.
The Hamiltonian for the two-band model is as follows [17]

$$\hat{H} = \begin{bmatrix} E_c(z) - e\phi(z,t) & \frac{p_{cv}}{m_e}\frac{\hbar}{i}\frac{\partial}{\partial z} \\ \frac{p_{vc}}{m_e}\frac{\hbar}{i}\frac{\partial}{\partial z} & E_v(z) - e\phi(z,t) \end{bmatrix} \tag{5}$$

where $\phi(z,t)$ is the scalar potential, $\mathrm{E}_c(z)$ is the conduction-band offset and $\mathrm{E}_v(z)$ approximates the valence-band offset. The stationary form without any external bias $\phi(z,t)$ reads

$$\begin{pmatrix} E_c(z) & \frac{p_{cv}}{m_e}\frac{\hbar}{i}\frac{\partial}{\partial z} \\ \frac{p_{vc}}{m_e}\frac{\hbar}{i}\frac{\partial}{\partial z} & E_v(z) \end{pmatrix} \begin{pmatrix} \Psi_c(z) \\ \Psi_v(z) \end{pmatrix} = E \begin{pmatrix} \Psi_c(z) \\ \Psi_v(z) \end{pmatrix} \tag{6}$$

Where the lower component reads

$$\frac{p_{vc}}{m_e}\frac{\hbar}{i}\frac{\partial \Psi_c(z)}{\partial z} + E_v(z)\Psi_v(z) = E\Psi_v(z) \tag{7}$$

Rearranging results in the expression for $\Psi_v(z)$ as follows

$$\Psi_v(z) = \frac{1}{E - E_v(z)}\frac{p_{vc}}{m_e}\frac{\hbar}{i}\frac{\partial \Psi_c(z)}{\partial z} \tag{8}$$

substituting this expression into the upper component of Eq. 6 the wavefunction for the conduction becomes

$$\left[E_c(v) - \frac{\partial}{\partial z}\frac{\hbar^2}{2m_c(E,z)}\frac{\partial}{\partial z}\right]\Psi_c(z) = E\Psi_c(z) \quad \text{with} \quad m_c(E,z) = \frac{m_e^2(E - E_v(z))}{2|p_{cv}|^2} \tag{9}$$

where $m_c(E,z)$ is the energy dependent effective mass, which also depends on z through the valence band offset, and $|p_{vc}|$ is the momentum operator obtained from the Kane Energy denoted

$$K = \frac{2|p_{vc}|^2}{m_e} \tag{10}$$

This value, p_{vc} is required to be constant throughout the heterostructure within the two band model. Using the known effective mass at the conduction-band offset $m(E_c, v)$ allows for the calculation of the valence band offset using

$$E_v(z) = E_c(z) - 2|p_{cv}|^2\frac{m_c^*(z)}{m_e^2} \tag{11}$$

The valence band value obtained through Eq. 11 does not agree with the material value, as it neglects the degeneracy of the valence band. Finally, by selecting the appropriate phase for the momentum matrix elements,

$$p_{vc} = i|p_{cv}| \tag{12}$$

a real Hamiltonian is obtained, enabling the possibility to select real eigenstates.

3.2 Bloch States

We now consider the periodicity in a QCL superlattice, with N layers of varying thickness of $b_1, b_2, ...b_N$, the Bloch conditions provides as follows

$$e^{iqd}\begin{pmatrix}\Psi_c^{q\nu}(z)\\ \Psi_v^{q\nu}(z)\end{pmatrix} = \begin{pmatrix}\Psi_c^{q\nu}(z+d)\\ \Psi_v^{q\nu}(z+d)\end{pmatrix} \tag{13}$$

where the Bloch states $(\Psi_c^{q\nu}(z), \Psi_v^{q\nu}(z))^{tr}$ are also the eigenstates for a vanishing potential $\phi(z)$ and d is the length of the module, $\Sigma_i b_i$. The Bloch states are further characterised by the Bloch vector q in the range of the first Brillouin zone $(-\pi/d < q \leq \pi/d)$, and the band index ν.

It is important to note that the effective mass for a given layer depends only on the energy and as the valence-band offset, $E_v(c)$, remains constant in the region $z_{i-1} < z_i < z_{i+1}$, the solution in this region of uniform potential is therefore given by the general solution to Schrodinger's equation

$$\Psi_c^i(z) = Ae^{\lambda_i(z-z_i)} + Be^{-\lambda_i(z-z_i)} \quad \text{where} \quad \lambda_i(E) = \frac{\sqrt{2m_i(E-E_c)}}{\hbar} \tag{14}$$

Ensuring that both $\Psi_c^i(z)$ and $\Psi_v^i(z)$ and their derivatives are continuous at the layer boundaries, that is to say

$$\Psi_c^i(z_i - 0^+) = \Psi_c^{i+1}(z_i + 0^+) \quad \text{and} \quad \frac{1}{m_i}\frac{\partial\Psi_c^i(z)}{\partial z}\bigg|_{z=z_i-0^+} = \frac{1}{m_{i+1}}\frac{\partial\Psi_c^{i+1}(z)}{\partial z}\bigg|_{z=z_i+0^+} \tag{15}$$

provides the relation between adjacent layers

$$\begin{pmatrix}A_{i+1}\\ B_{i+1}\end{pmatrix} = \mathcal{M}_i\begin{pmatrix}A_i\\ B_i\end{pmatrix} \quad \text{with} \quad \mathcal{M}_i = \begin{bmatrix}e^{\lambda_i b_i}(1+\alpha_n) & e^{-\lambda_i b_i}(1-\alpha_n)\\ e^{\lambda_i b_i}(1-\alpha_n) & e^{-\lambda_i b_i}(1+\alpha_n)\end{bmatrix} \tag{16}$$

where $\alpha_i = \frac{m_{i+1}\lambda_i}{m_i\lambda_{i+1}}$ and the matrix $\mathcal{M}_i$ is energy-dependent through λ_i and α_i, which may also be complex in the instance that $E > E_c(z)$.
A further boundary condition may be applied at the end of the module to satisfy the Bloch condition in Eq. 13. With N being the number of layers in each module, we obtain the following Bloch relation between modules.

$$e^{iqd}\begin{pmatrix}A_1\\ B_1\end{pmatrix} = \begin{pmatrix}A_{N+1}\\ B_{N+1}\end{pmatrix} = \mathcal{M}\begin{pmatrix}A_1\\ B_1\end{pmatrix} \text{ with } \quad \mathcal{M} = \mathcal{M}_N\mathcal{M}_{N-1}\mathcal{M}_{N-2}...\mathcal{M}_1 \tag{17}$$

Given this condition A_1 and B_1 are non-vanishing only if $\det\{\mathcal{M}_N — e^{iqd}\mathrm{I}\} = 0$, resulting in a set of solutions $E_\nu(q)$ for a given Bloch vector q. From these energies

$E_\nu(q)$ one may evaluate the corresponding eigenvectors $(A_1, B_1)^{tr}$ and subsequently the whole set of wavefunction coeffiecients using Eq. 16

$$\begin{pmatrix} A_i \\ B_i \end{pmatrix} = \mathcal{M}_i ... \mathcal{M}_2 \mathcal{M}_1 \begin{pmatrix} A_1 \\ B_1 \end{pmatrix}$$

These coefficients then enable the construction of the wavefunction for the conduction band for an arbitrary z using Eq. 14. From these evaluated wavefunctions $\Psi_c^i(z)$, the valence band wavefunctions may be determined via Eq. 8. All that is now left is to normalise the obtained wavefunctions, this is performed by setting the integral over the entire module of the sum of the absolute wavefunctions squared to one.

$$\int_0^d [|\Psi_c^{\nu q}(z)|^2 + |\Psi_v^{\nu q}(z)|^2]\, dz = 1 \tag{18}$$

3.3 Wannier States

So far, we have calculated the Bloch functions for an arbitrary complex phase. The localised Wannier functions are given by

$$\begin{pmatrix} \omega_c^{\nu i}(z) \\ \omega_v^{\nu i}(z) \end{pmatrix} = \frac{d}{2\pi} \int_{-\pi/d}^{\pi/d} dq\, \mathrm{e}^{-iqnd} \begin{pmatrix} \Psi_c^{q\nu}(z) \\ \Psi_v^{q\nu}(z) \end{pmatrix} e^{i\phi q} \approx \frac{1}{j} \sum_j \mathrm{e}^{-iq_j nd} \begin{pmatrix} \Psi_c^{q_j \nu}(z) \\ \Psi_v^{q_j \nu}(z) \end{pmatrix} e^{i\phi q} \tag{19}$$

In order to evaluate the localised Wannier functions, one must choose phase factor which is q-dependant, $e^{i\phi_q}$, with the constraint $\phi_{-q} = -\phi_q$. In 1959, Kohn demonstrated that selecting Wannier functions to be real at symmetry points often yields the most optimal localization [11]. In the case of quantum cascade lasers (QCLs), such symmetry points are non-existent, making the selection process more challenging. However, a viable strategy is to choose all Bloch states to be real and positive at specific points. This approach often proves advantageous, leading to Wannier functions that are localized at the chosen artificial symmetry points.
To calculate the phases we adopt the method outlined in [3], which necessitates selecting initial phases in such a way that the Bloch functions remain continuous in q, accounting for the periodicity at $q = \pm\pi/d$. The phase is then given by Eq. 20 of [3],

$$\phi_\nu^{ML}(q) = \int_0^q dq' [X_\nu(q') - x_\nu] \tag{20}$$

where $X_\nu(q')$ snd x_ν are taken from Eq. 12 and 13 respectively, denoted

$$x_\nu = \frac{d}{2\pi}\int_{-\pi/d}^{\pi/d} dq X_\nu(q), \quad X_\nu(q) = \mathrm{i}\int_0^d dz e^{iqz}((\Psi_c^{q\nu}(z))(\Psi_v^{q\nu}(z))^* \frac{\partial}{\partial q}\left[e^{iqz}\begin{pmatrix}\Psi_c^{q\nu}(z)\\ \Psi_v^{q\nu}(z)\end{pmatrix}\right]$$

The Bloch states with index ν, q are then multiplied by their corresponding phase factors. It is these states that are employed in Eq. 16 to calculate the maximally localized Wannier functions.

4 Implementation of Localised Wannier Functions

4.1 Method

To compute the localized Wannier functions the necessary parameters for a given heterostructure must be provided. Namely the thickness of each layer b_i, the conduction band offset m_c^*, which represents the barrier height, the effective mass at the conduction band offset energy E_c, also known as the Γ-point and the Kane Energy. The algorithm requires the inputs to be provided as numpy arrays in order to perform subsequent calculations. Calculations begin by identifying the energy eigenvalues that fulfill the Bloch condition. In other words, that is to solve $f(E) = \det\{\mathcal{M}_N - e^{iqd}\mathrm{I}\} = 0$

The old code achieved this by the use of the Bisection method. This method iteratively narrows down the search interval of a given function by dividing it in half and selecting the sub-interval in which the function changes sign, effectively converging towards the root, as depicted in Fig.4.

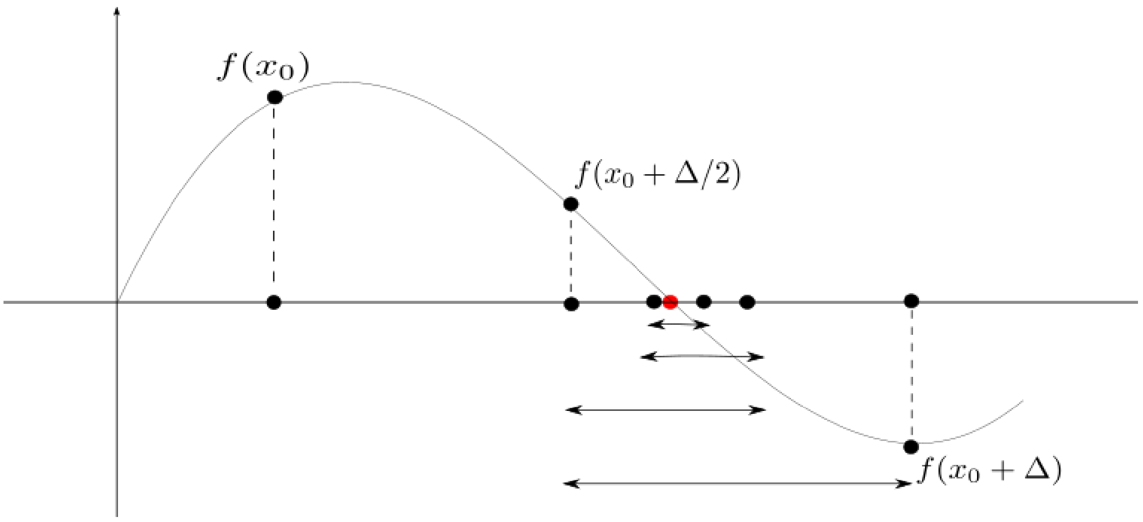

Figure 4: For an interval $\{x, x + \Delta\}$, the method works by finding the function sign at the mid-point $f(x + \Delta/2)$, subsequently narrowing the search interval

However, when dealing with a large number of roots, this method can become time-consuming due to the need for iterative evaluations of the function and updating of interval bounds.

To improve efficiency, the new code incorporated the root_scalar function from the scipy.optimize library, which leverages faster programming languages like C, C++, and Fortran for time-critical computational loops. An additional challenge arose from the fact that the function $f(E)$ has multiple roots, in fact $\lim_{E\to\infty} f(E) = 0$.

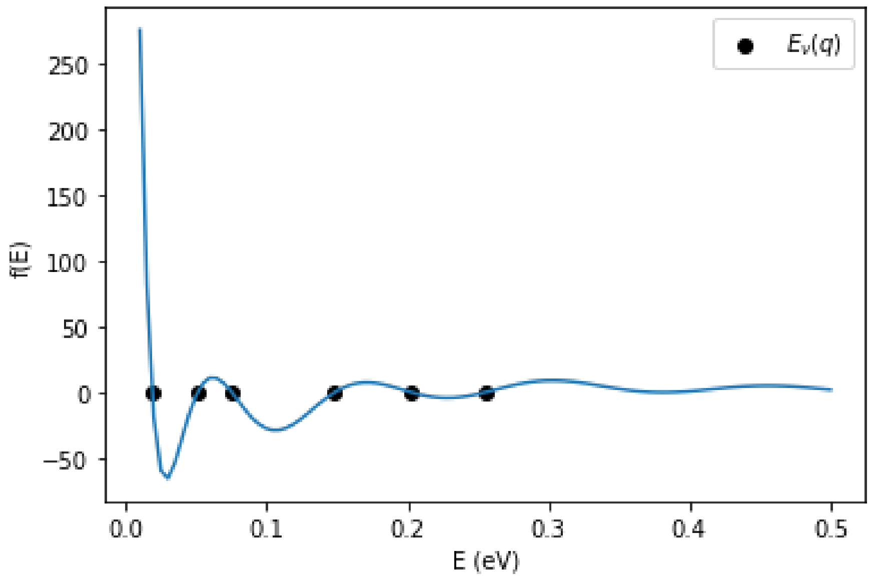

Figure 5: As seen, $f(E)$ oscillates and dampens around the x-axis, proving difficult for the root_scalar function to solve over an unspecified interval

This posed difficulties for root_scalar without an appropriate search interval. To address this, a hybrid approach was devised, combining a modified bisection method with root_scalar. This hybrid method starts with an initial guess and step size to identify the interval where the root is likely to lie, by checking for a sign change. Finally, root_scalar is used to converge on the root itself. By combining the strengths of both methods, the code achieves a more efficient and accurate determination of the energy roots compared to relying solely on either method.

For each wavevector q the number of roots found, N_r, is an initial parameter set, each root is then identified by its band index and q, $E(\nu, q)$. It is important to discard any energies higher than the barrier height, since these states cannot easily be localised within the structure.

The q-values used to determine the Bloch functions are chosen such that they cover the first Brillouin zone with equal spacing.

$$q_j = -\frac{\pi}{d} - \frac{\pi}{N_q d} + \frac{2\pi}{N_q d} j \quad \text{for} \quad j = 1, 2, 3....N_q$$

These q_j values restricts to the eigenstates of a system of length $L = N_q d$, with boundary conditions $\Psi(x+L) = (-1)^{N_q+1}\Psi(x)$, resulting in the Born von-Karman boundary condition for odd N_q. After the eigenvalues $E(\nu, q)$ have been determined, the eigenvectors A_1 and B_1 are evaluated as

$$A_1 = -\epsilon_{11},\ B_1 = \epsilon_{12} \tag{21}$$

where ϵ_{11} and ϵ_{12} are given by

$$\mathcal{M} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{21} & \epsilon_{22} \end{bmatrix} \quad \text{for} \quad \mathcal{M} = \mathcal{M}_N \ — \ e^{iqd}\mathrm{I}$$

Subsequent co-efficients $[A_i, B_i]$ are determined using Eq. 16, allowing for the construction of the conduction band Bloch wavefunctions for a given position z using Eq. 14. The valence band functions $\Psi_v^i(z)$ are further obtained via Eq. 8. Both functions are then normalised by re-scaling A_n and B_n to satisfy Eq. 18. All that remains is to determine the phase factor from Eq. 20 and evaluating the Wannier function using Eq. 19.

4.2 Special Techniques

In the calculation of the Bloch functions, certain specialized techniques were employed to perform specific computations. One notable technique was the utilization of numpy.roll for calculating function derivatives. By appropriately shifting the array elements, the derivative of a function can be calculated using finite differences. This approach proves useful in situations where analytical derivatives are not readily available. Additionally, masks were employed to generate functions for different regions. A mask is a binary array that serves as a filter to select specific elements or regions of an array. By applying masks, one can selectively include or exclude elements based on specified conditions. In this particular context, masks were employed to delineate and define distinct layer regions for the calculation of their respective Bloch functions. Thus specific areas within the system could be identified and isolated, allowing for targeted computations on individual layers.

5 Simulated QCL Structures

The table below illustrates the structures of THz QCL models in this study, showcasing the layer sequences and corresponding materials. The structures G938 and G936 taken from [14] were chosen due to their similarity to the model LU2022 which the code was initially modelled for.

Label	x	Layer Sequence (nm)	Material
LU2022	0.3	**3.1**, 7.1, **2.1**, 14.2	$\mathbf{Al_xGa_{1-x}As}$, GaAs
G938	0.35	**2.88**, 7.45, **1.76**, 15	$\mathbf{Al_xGa_{1-x}As}$, GaAs
G936	0.3	**3.3**, 6.43, **1.9**, 14.7, **2.4**, 8.33	$\mathbf{Al_xGa_{1-x}As}$, GaAs

Table 1: Table of structures of simulated models from [9], where the bold layers denote the barriers separating the GaAs wells.

5.1 Under Zero Bias

The Longitudinal Optical Phonon Energy of GaAs is ∼36.6meV [14], The following three simulated structure are very similar, differing omly in the Al composition in the barriers in the case of G938, and the number of layers in the case of G936.

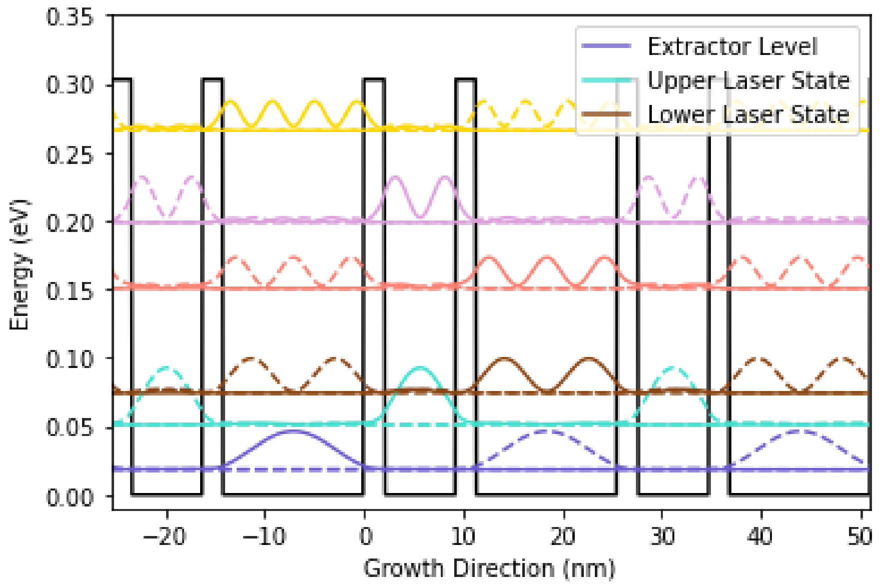

Figure 6: LU2022 structure under no bias, here the energy states of interest are the three first states.

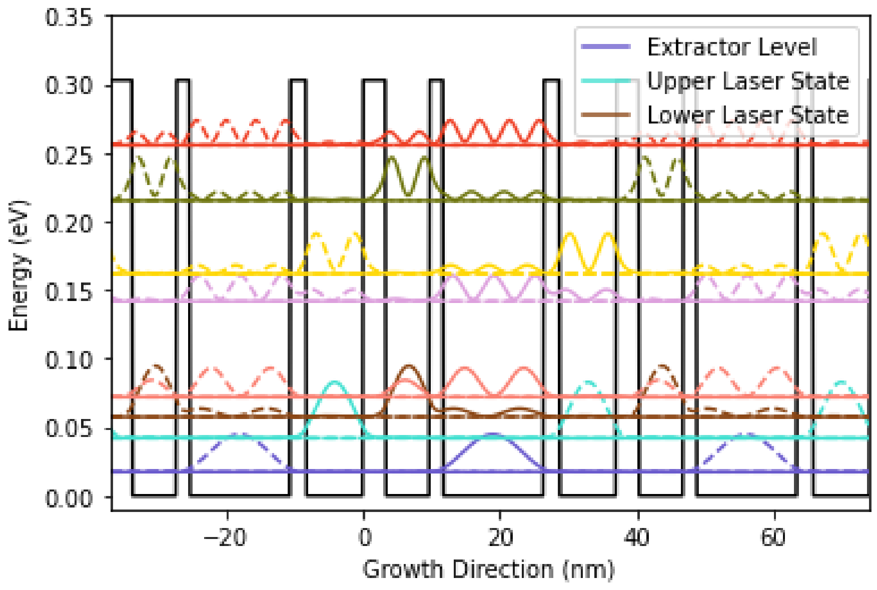

Figure 7: Model G936 similar to LU2022 with an added well and barrier providing additional energy states denoted by the turquoise and yellow bands.

In Figure 7 the 2^{nd} excited state and the extractor state are no longer spatially aligned, rendering the 3^{rd} excited state to now act as the LLS.

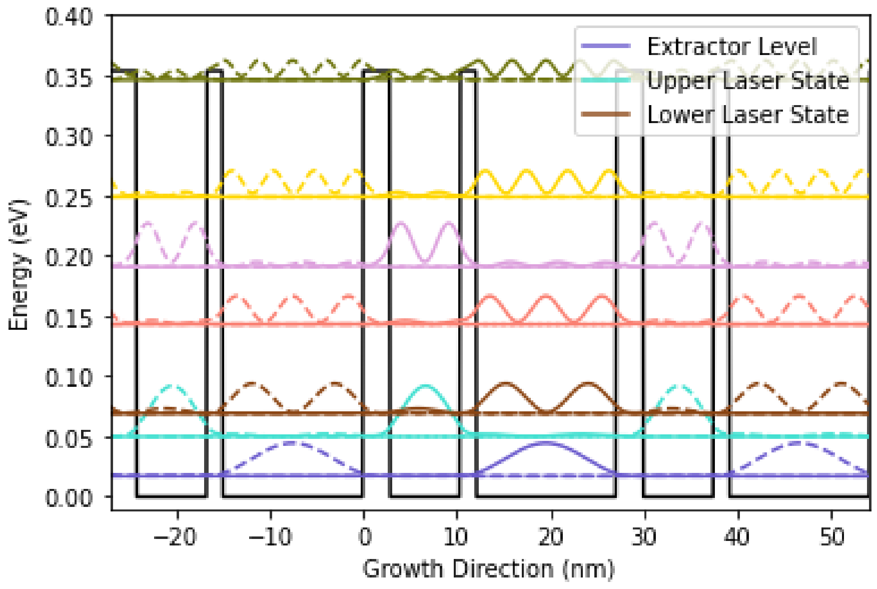

Figure 8: G938 with an increased barrier height, here the barrier widths are smaller, increasing resonant tunnelling rate from the extractor to the ULS.

5.2 Under Bias

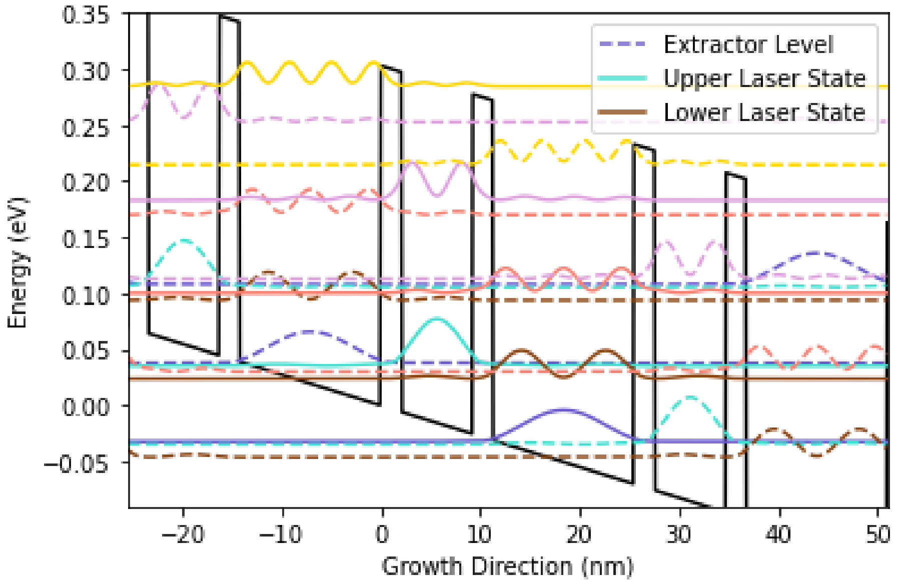

Figure 9: LU2022, under 0.07eV bias per module, extraction and injection are performed by DP and RT respectively.

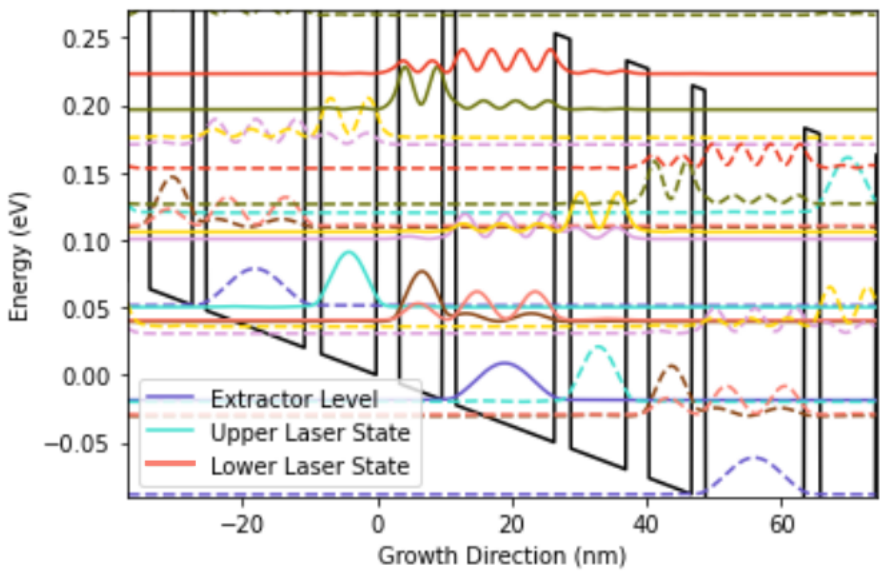

Figure 10: G936 under same bias as LU2022, the lower laser state is now the 3^{rd} excited stated, since it spatially alligns with extractor state.

The bias 0.07meV was chosen in order that the ULS and the Extractor state are resonant, meaning that the two states align energetically, thus increasing the tunnelling amplitude through the barrier. Fig. 9 & Fig. 10 represent structures with similar barrier heights, however in the case of the latter, the 2^{nd} excited state which was previously the LLS no longer spatially aligns with the extractor state. For this reason the the 3^{rd} excited becomes the LLS. This key observation highlights the flexibility and customisability of QCLs and the importance of simulation frameworks in identifying the roles each energy state plays.

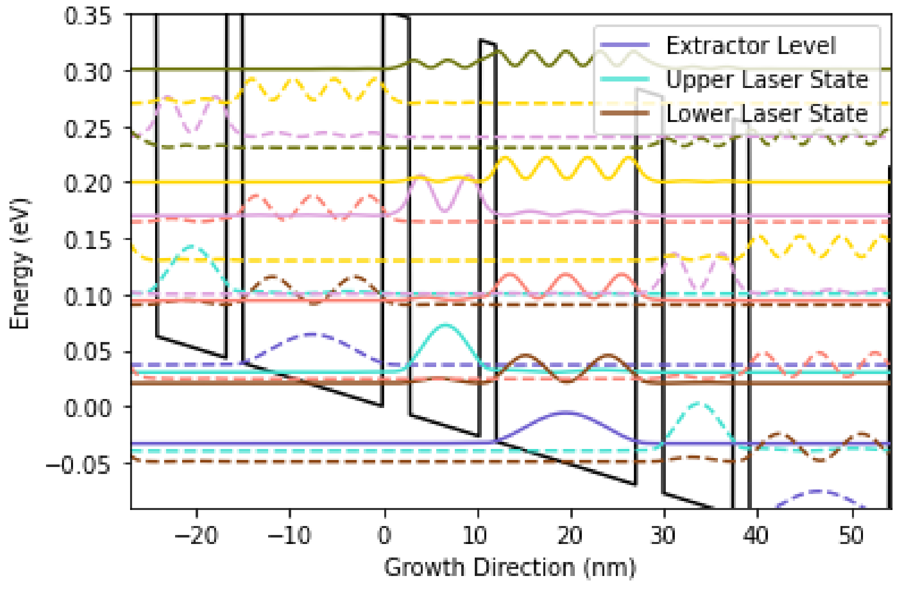

Figure 11: G938 again under 0.07meV bias per module, here similar states play similar roles to LU2022, with an expected increase in resonant tunneling due to decrease in barrier widths.

Larger barrier heights allow for more localised states, which can be beneficial for QCLs operating at higher energies, i.e. MID-IR QCLs, but may not be so productive for THz QCLs such as G938 model shown in Fig. 11. This due to the overlapping of the energy state since they are in such close proximity, creating a lot of noise since the electron may now tunnel into undesired states. This is also one of the reasons why THz QCL require to be at cryogenic temperatures to operate.

6 Conclusion and Outlook

Simulations and modeling play a pivotal role in the development and understanding of Quantum Cascade Lasers. These tools allow researchers to investigate the physical behaviors of QCL devices, resulting in optimized performances, and gaining insights that are otherwise challenging to obtain experimentally. In this thesis, the focus was on enhancing the existing Fortran code by developing a Python framework, which offers increased readability and flexibility, thereby establishing a foundation for more sophisticated calculations. The developed code is not yet complete and requires further testing and improvement, in particular, specific anomaly situations, such as when the energy is equivalent to the barrier height, need to be examined more carefully. In conclusion, despite the minor issues that exist in the developed code, it serves as a solid foundation for conducting further complex calculations in Quantum Cascade Lasers research. With this improved framework, researchers can now embark on improving other aspects related to QCLs, such as gain and operating temperatures, thus unlocking their full potential for a wide range of applications.

References

[1] Mattias Beck, Daniel Hofstetter, Thierry Aellen, Jérôme Faist, Ursula Oesterle, Marc Ilegems, Emilio Gini, and Hans Melchior. Continuous wave operation of a mid-infrared semiconductor laser at room temperature. *Science*, 295(5553):301–305, 2002.

[2] Felix Bloch. Über die quantenmechanik der elektronen in kristallgittern. *Zeitschrift für physik*, 52(7-8):555–600, 1929.

[3] Alexys Bruno-Alfonso and Dennis R. Nacbar. Wannier functions of isolated bands in one-dimensional crystals. *Physical Review B*, 75(11):115428, 2007.

[4] Raymond Dingle and Charles H. Henry. Quantum effects in heterostructure lasers, September 21 1976. US Patent 3,982,207.

[5] Leo Esaki and Raphael Tsu. Superlattice and negative differential conductivity in semiconductors. *IBM Journal of Research and Development*, 14(1):61–65, 1970.

[6] Jerome Faist, Federico Capasso, Deborah L. Sivco, Carlo Sirtori, Albert L. Hutchinson, and Alfred Y. Cho. Quantum cascade laser. *Science*, 264(5158):553–556, 1994.

[7] Claire Gmachl, Federico Capasso, Deborah L. Sivco, and Alfred Y. Cho. Recent progress in quantum cascade lasers and applications. *Reports on progress in physics*, 64(11):1533, 2001.

[8] Christian Jirauschek and Tillmann Kubis. Modeling techniques for quantum cascade lasers. *Applied Physics Reviews*, 1, 02 2014.

[9] Ali Khalatpour, Man C. Tam, Sadhvikas J. Addamane, John L. Reno, Zbignew Wasilewski, and Qing Hu. Enhanced operating temperature in terahertz quantum cascade lasers based on direct phonon depopulation. *Applied Physics Letters*, 122(16):161101, 2023.

[10] Sudeep Khanal, John L. Reno, and Sushil Kumar. 2.1 thz quantum-cascade laser operating up to 144 k based on a scattering-assisted injection design. *Optics express*, 23(15):19689–19697, 2015.

[11] Walter Kohn. Analytic properties of bloch waves and wannier functions. *Physical Review*, 115(4):809, 1959.

[12] Sushil Kumar, Chun Wang I. Chan, Qing Hu, and John L. Reno. Two-well terahertz quantum-cascade laser with direct intrawell-phonon depopulation. *Applied Physics Letters*, 95(14):141110, 2009.

[13] Mark Lee and Michael C. Wanke. Searching for a solid-state terahertz technology. *Science*, 316(5821):64–65, 2007.

[14] Denizhan E. Önder. Dynamical analysis of terahertz quantum cascade lasers. 2023.

[15] Carlo Sirtori, Federico Capasso, Jérôme Faist, and Sandro Scandolo. Nonparabolicity and a sum rule associated with bound-to-bound and bound-to-continuum intersubband transitions in quantum wells. *Physical Review B*, 50(12):8663, 1994.

[16] Gregory H. Wannier. The structure of electronic excitation levels in insulating crystals. *Physical Review*, 52(3):191, 1937.

[17] Steven R. White and Lu J. Sham. Electronic properties of flat-band semiconductor heterostructures. *Physical Review Letters*, 47(12):879, 1981.

[18] Benjamin S. Williams, Sushil Kumar, Hans Callebaut, Qing Hu, and John L. Reno. Terahertz quantum-cascade laser operating up to 137 k. *Applied Physics Letters*, 83(25):5142–5144, 2003.

7 Appendix

```
"""
Created on Tue Feb 21 18:34:42 2023

@author: zakariam
"""
import numpy as np
from scipy.optimize import root_scalar
import scipy.constants as cd
import matplotlib.pyplot as plt
import pandas as pd

#%%
# constants

c = cd.e # electric charge
me = cd.m_e# mass of electron
hbar = cd.hbar
C = np.sqrt(me*c)* 1e-9 / hbar

#%%

#input variables, in final version these will be empty lists, appended with values from an input file

thlist = np.array([3.1, 7.1, 2.1, 14.2 ]) # thickness of the layers
Ecn = np.array([0.303, 0, 0.303, 0])# conduction band offset
mcn = np.array([0.0919, 0.067, 0.0919, 0.067])# eff mass, in me
p = 22.67# Eg/meff
d = sum(thlist)# 1e-9 # thickness of module
Nq = 900 #Number of q values
Ev = Ecn - (p*mcn)#Valence band offset
Nr = 8#Number of bands
Nz = 300

```

```

#%%
def lambdar(E):# function to calculate the effective
    mass and lambda for each layer
    m = (E- Ev)/p
    l = np.sqrt(2*m*(Ecn - E) + 0j)*C
    return l, m


def layer_matrices(E):#function to calculate
   hamiltonian the matrices for each layer
    l, m = lambdar(E)
    a = (np.roll(m,1)/m) * (l/np.roll(l,1))#
       expression for alpha
    b = thlist
    exp1 = np.exp(l*b)
    exp2 = 1/exp1
    M = 0.5 * np.array([[exp1*(1 + a), exp2*(1 - a)
       ], #matrix of current layer
                         [exp1*(1 - a), exp2*(1 + a
                           )]])

    M = np.rollaxis(M, 2)#shift axis of the matrix
       into a more understandble form

    return M#returns H-matrix for each layer


#%%

def moduleprod(E):
    md = layer_matrices(E)
    result = md[0]
    for matrix in md[1:]:# iterate over matrices,
       except first matrix
        result = np.matmul(matrix, result)# multiply
            matrices
    return result#return product(matrix of final
       layer)

```

```python

def det_func(E, q):#def det_func(E):# the determinant that must be zeroed
    return np.linalg.det(moduleprod(E) - np.eye(2)*np.exp(1j*q*d))


def rootfinder(q):
    # Define the bounds of the energy search interval
    E_min = 0.001
    E_max = 0.011

    # Find all the roots using root_scalar
    roots = []
    while True:
        if np.sign(det_func(E_min, q)) != np.sign(det_func(E_max, q)):
            sol = root_scalar(det_func, bracket=[E_min, E_max], args=(q,), method='brentq')
            roots.append(sol.root)
            E_min = sol.root + 0.01
            E_max = E_min + 0.01
        else:
            E_min = E_max
            E_max += 0.01

        # Stop if the maximum number of roots is found or if the search interval is too large
        if len(roots) >= Nr:
            break

    return roots


#q = np.linspace(-np.pi/d, np.pi/d, Nq)# Define the range of q values
```

```python
q = np.linspace(-np.pi/d*(1-1/Nq), np.pi/d*(1-1/Nq),
    Nq)# Define the range of q values as script NEW


def rootsarr(Nq):#function to store the roots for
   each q value
    roots_arr = np.zeros((Nq, Nr, 3), dtype=np.
       complex128) # Initialize an array to store
       the roots for each q value

    for i in range(Nq):
        roots = rootfinder(q[i])# Find the roots for
            the current q value
        for j in range(Nr):

            M = moduleprod(roots[j])# Calculate the
               module product for the current root

            # Obtain co-effiecients A and B from
               current matrix
            A = -M[0,1]
            B = M[0,0] - np.exp(1j*q[i]*d)
            #print(B)

            # Store the root and its associated
               values in the roots_arr array
            roots_arr[i,j,:] = np.array([roots[j], A
               , B])

    return roots_arr

# df is numpy array [iq,inu,sort] sort=0:E sort=1:A,
   sort=2:B
df = rootsarr(Nq)#make roots_arr global for ease of
   access to values


#%%

```

```python
def psi_Cfunc(A, B, lambd, z_n, z_arr):#function to construct conduction band wavefunction
    exper = np.exp(lambd * (z_arr - z_n))
    #here z_n-z_arr gives the distance travelled within a specific layer
    return A*exper + B/exper

def psiV_func(E, Ev, grad):#function to calculate valence band wavefunction
    const = np.sqrt(p*me*c/2)*(hbar/me)*1e9
    return (1/(E - Ev))*(const/c)*grad


z = np.linspace(0, d, 300, endpoint=False)
dz = z[1]-z[0]


def derivative_arr(i, arr):#bespoke derivative calculator with the correct Bloch boundary conditions
    del_arr = (np.roll(arr, -1) - np.roll(arr, 1))/(2*dz)
    del_arr[0] = (arr[1] - np.exp(-1j*q[i]*d)*arr[-1])/ (2*dz)
    del_arr[-1] = (np.exp(1j*q[i]*d)*arr[0]-arr[-2]) / (2*dz) #NEW

    return del_arr


def calculate_Bloch(q, v):
    # Define the boundaries of each layer
    bounds = np.zeros(len(thlist)+1)
    bounds[1:] = np.cumsum(thlist)

    # Initialize arrays for the conduction and valence band wavefunctions
    psiC_arr = np.zeros(len(z), dtype=np.complex128)
    psiV_arr = np.zeros(len(z), dtype=np.complex128)

    # Get the coefficients for the current energy eigenvalue and wavevector
    coeff = df[q,v,:]
    E = coeff[0]
    AB = coeff[1:].reshape((2,1))

    # Obtain Lamba and layer matricies for the given energy
    lambd, _ = lambdar(E)
    M = layer_matrices(E)

    ev_arr = np.zeros(len(z))
    for i in range(len(bounds)-1):
        # Mask to select the points in the current layer
        mask = (z >= bounds[i]) & (z < bounds[i+1])
        ev_arr[mask]=Ev[i]   #NEW

        # Calculate the conduction band wavefunction for the current layer
        psiC_arr[mask] = psi_Cfunc(AB[0], AB[1], lambd[i], bounds[i], z[mask])

        # Update the coefficients for the next layer
        AB = M[i] @ AB

    # Calculate the derivative of the conduction band wavefunction
    del_arr = derivative_arr(q, psiC_arr)

    # Calculate the valence band wavefunction for each point in the z-direction
    for x in range(len(z)):
        psiV_arr[x] = psiV_func(E, ev_arr[x], del_arr[x])
        #psiV_arr.append(psiV_func(E, ev_arr[x], del_arr[x]))

    # Convert the list of valence band wavefunction values to a numpy array
```

```python
    #psiV_arr = np.array(psiV_arr, dtype=np.complex128)

    # Normalize the wavefunctions
    norm_const = sum((abs(psiC_arr)**2 + abs(psiV_arr)**2)*dz)
    psiC_arr, psiV_arr = psiC_arr/np.sqrt(norm_const), psiV_arr/np.sqrt(norm_const)

    # Return the normalised conduction and valence wave-functions array
    return np.array([psiC_arr, psiV_arr])


def Total_Bloch(v):
    Bloch_arr = np.zeros((Nq, 2, Nz), dtype = np.complex128) #initialize array to hold the Bloch wavefunctions for all q NEW

    for f in range(Nq):# loop over all q-values and calculate the Bloch wavefunctions for a given band v
        Bloch_arr[f,:] = calculate_Bloch(f,v)

    return Bloch_arr

def BlochPhase(arr):
    Xarr = np.zeros(Nq, dtype=np.complex128)
    arr1 = np.exp(-1j*np.outer(q, z))
    Harr = arr*np.expand_dims(arr1, axis=1)
    delarr = (np.roll(arr, -1, axis=0)*np.expand_dims(np.exp(-1j*np.outer(np.roll(q, -1), z)), axis=1)
               - np.roll(arr, 1, axis=0)*np.expand_dims(np.exp(-1j*np.outer(np.roll(q, 1), z)), axis=1))*Nq*d/(4*np.pi)
    delarr[0,:,:] = ((arr[1,:,:]*np.exp(-1j*q[1]*z))
                          - (arr[-1,:,:]* np.exp(-1j*(q[-1] - (2*np.pi/d))*
                             z)))*Nq*d/2/np.pi
    delarr[-1,:,:] = (arr[0,:,:]*np.exp(-1j*(q[0]+(2*np.pi/d))*z[:])
                          -arr[-2,:,:]*np.exp(-1j*q[-2]*z[:]))*Nq*d/2/np.pi



    Xarr = 1j*dz*np.sum(np.sum(np.conjugate(Harr)*delarr, axis=1), axis=1)
    xv = (np.sum(Xarr, axis=0))/Nq
    sX = Xarr - xv
    sX = np.roll(sX, int(Nq/2))
    phase = np.cumsum(sX)*2*np.pi/(Nq*d)

    return np.roll(phase, -int(Nq/2))


def fullbloch(n, v):
    arr = Total_Bloch(v)
    zmax = np.linspace(-n*d, (n+1)*d, (2*n+1)*Nz, endpoint=False) #NEW
    factor = np.arange(-n, n+1)
    factor = np.outer(q, factor)
    factor = np.exp(factor*1j*d)
    factor = np.repeat(factor, Nz, axis=1)
    factor = np.expand_dims(factor, axis=1)

    arr = np.tile(arr, (1, 1, 2*n+1))
    full_bloch = arr * factor
    return full_bloch
#%%

def expectationvaluez(n, v):
    FullBloch = fullbloch(n, v)
    Bloch =Total_Bloch(v)
    zmax = np.linspace(-n*d, (n+1)*d, (2*n+1)*Nz, endpoint=False)
    phase = BlochPhase(Bloch)
```

```python
    phasearr = np.tile(np.exp(1j*phase), (Nz*(2*n+1), 1)).transpose()
    FullBloch *= np.expand_dims(phasearr, axis=1)


    wannier_func = (np.sum(FullBloch, axis=0))/Nq
    absWannier =  abs(wannier_func[0])**2+abs(wannier_func[1])**2

    #norm=sum(absWannier)*dz
    ez =sum(absWannier*zmax)*dz

    return ez

for i in range(Nr):
    print(expectationvaluez(3, i))

def wannier(v):
    arr = Total_Bloch(v)
    phase = BlochPhase(arr)
    phase = np.exp(1j*phase)
    arr *= phase[:, np.newaxis, np.newaxis]
    wannier = (np.sum(arr, axis=0))/Nq

    return wannier

def fullwannier(n, v):
    fullarr = fullbloch(n, v)
    arr = Total_Bloch(v)
    phase = BlochPhase(arr)
    phase = np.exp(1j*phase)
    fullarr *= phase[:, np.newaxis, np.newaxis]
    fullwannier = (np.sum(fullarr, axis=0))/Nq

    return fullwannier
```