SUNY Institute of Technology

The Kronig-Penney Model:

Finding the Eigenvalues and Wavefunctions of a Particle in a Crystal Lattice

Written by:

Daniel Landers

Major: Electrical and Computer Engineering

Concentrations: Electrical Engineering and Physics

Prepared for

Instructor: Dr. A. H. Fariborz

Course: PHY 381 – Quantum Mechanics

Semester: Spring 2014

Abstract:

The Kronig-Penney model is examined, which addresses the problem of finding the eigenvalues of a particle in a one-dimensional periodic crystal lattice. In the Kronig-Penney model the true periodic potential of the particle and lattice, is replaced with square periodic potential wells. The eigenvalues within and above the potential well are derived. There can also be an eigenvalue of zero, which is the upper threshold of the potential well and exists in special conditions that are derived herein. The conditions for eigenvalues are depicted along with the bands of allowed energies and the forbidden gaps. The corresponding general wavefunctions of the particle are also revealed. Complications arise when attempting to normalize the wavefunctions. Nonetheless, examples of possible normalized wavefunctions are provided, and the subsequent probability density functions are truly representative of the model.

Part 1: Introduction

1.1: A Particle in a One-Dimensional Crystal Lattice

In a crystal lattice there is a periodic structure of positively-charged ions. A negatively charged particle, like an electron, would experience an attractive electromagnetic force towards the positive ions. Its potential energy in one dimension is shown figure 1A. The location of the positive ions coincides with the potential valleys. Finding the eigenvalues of this potential would be no easy feat. If we replace the periodic valleys with finite square wells as in figure 1B, then we can quickly find the solution to the Schrödinger equation within and outside the potential wells. This method provides a good approximation of the eigenvalues and is known as the Kronig-Penney Model.

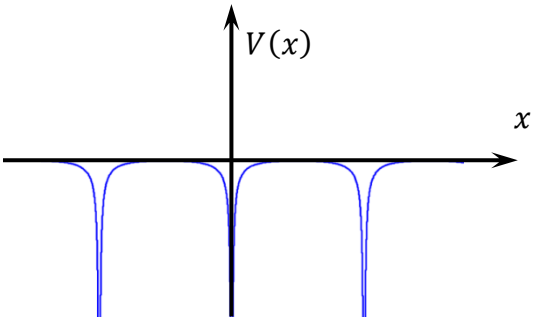

Figure 1A: Realistic Potential Function

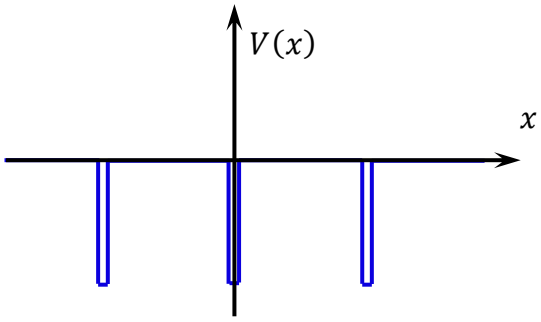

Figure 1B: Kronig-Penney Model

In this project we'll find the general solutions to the Schrodinger equation in all regions of the Kronig-Penney model's periodic potential in figure 1B. This will lead to the eigenvalues of the system, which are dependent on the dimensions of the periodic potential. The conditions for eigenvalues will be shown, along with the bands of possible eigenvalues and the forbidden gaps. Also, possible wavefunctions will be depicted along with their probability density functions. There were some practical complications in normalizing the wavefunctions, particularly in obtaining their phase, but the resulting probability density functions are truly representative of the system

In this system, the wavefunctions can take on different forms, depending on whether the particle is bound in the potential well, with a negative eigenvalue, or unbound with a positive eigenvalue. Therefore we'll find the conditions for both the positive and negative eigenvalues. Under special conditions there can exist an eigenvalue at zero. We'll ascertain whether the conditions for positive, negative and the zero eigenvalue are related, equivalent, or simply unique.

Before attempting to find the eigenvalues and eigenfunctions, we need to elaborate on the characteristics of the crystal lattice. Finding the wavefunction in every period of the crystal could be a burdensome task as the crystal becomes increasingly large. Fortunately, we can solve Schrodinger's equation fully and compactly in a periodic potential, by invoking Bloch's Theorem.

1.2: Bloch's Theorem

A potential energy function that is periodic repeats itself as:

$$V(x+d) = V(x). \tag{1}$$

Bloch's Theorem states that for periodic potentials, the solutions of the time-independent Schrödinger equation,

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi = E\psi, \tag{2}$$

are of the form

$$\psi(x+d) = \psi(x)e^{i\mathrm{K}d}. \tag{3}$$

If we allowed our periodic potential to go on infinitely, in x, we could not obtain a normalizable wavefunction. Also, if the crystal structure were to have an edge, the potential would no longer be periodic outside of the crystal. So for Bloch's Theorem to be useful we need to wrap the x-axis of the potential around itself in a circle so that after N periods the function returns back to $x = 0$, so that

$$\psi(x+Nd) = \psi(x). \tag{4}$$

While this abstraction is unrealistic, the effects of the non-periodic edges of a crystal are approximately nonexistent, deep inside the crystal. Therefore we will assume that the potential and the wavefunction are perfectly periodic. From equations (3) and (4) we find that

$$e^{iNKd}\psi(x) = \psi(x). \tag{5}$$

Which leads to the definition of the constant K:

$$K = \frac{2\pi n}{Nd}, \text{where } n = 0, \pm 1, 2, 3, \ldots \tag{6}$$

N is the number of periods in the crystal. n is called the propagation constant, which along with N determines the allowed phases of the complex exponential term: $e^{i\mathrm{K}d}$. n may be incremented infinitely, but once it passes N, the phases repeat.

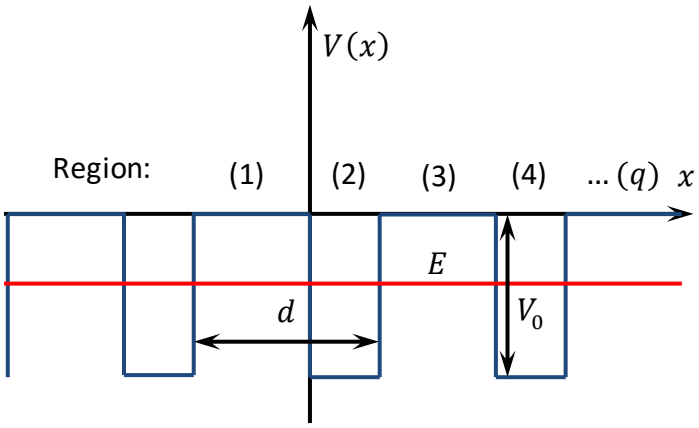

Figure 2: A Periodic Potential Energy Function: $V(x)$

Here's an example of a periodic potential to illustrate the use of Bloch's Theorem to find the wavefunction of an arbitrary region, q. If we shift by p periods (or $2p$ regions) we can represent the wavefunction in the qth region in terms of a wavefunction in a reference region, r:

$$\psi_q(x) = \psi_{r+2p}(x) = \psi_r(x - pd)e^{ip\mathrm{K}d}, \tag{7}$$

where $r, p,$ and q are integers. r is the number of the reference region, and p is the number of periods shifted to arrive at the qth region, whose wavefunction is in question. In the current scenario 1 and 2 are ultimately the reference regions. For example:

$$\psi_3(x) = \psi_1(x - d)e^{i\mathrm{K}d}. \tag{8}$$

Part 2: Negative Total Energy: $E < 0$

In this section we'll find the negative allowed energies. Assuming negative energies, we'll find the general solution to the Schrödinger equation for the Kronig-Penney model. Then we'll find the bands of allowed energies and forbidden gaps, and some possible wavefunctions.

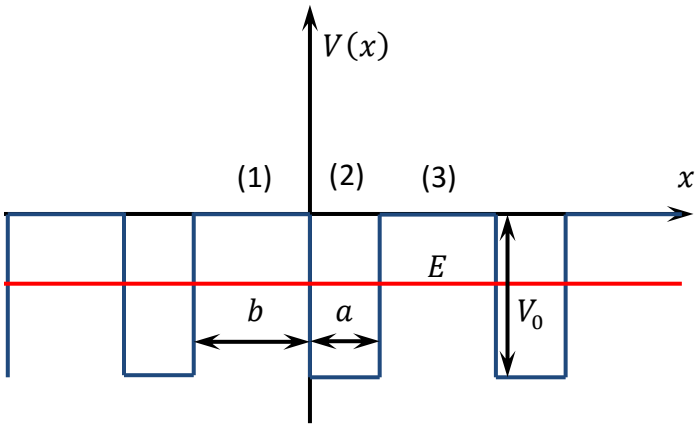

Figure 3: Kronig-Penny Model with Negative Total Energy: E

2.1: The Wavefunction within Regions 1 and 2: ψ_1^-, ψ_2^-:

In region 1 $(-b < x < 0)$, where the potential is zero, the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_1^-}{dx^2} = E\psi_1^-. \tag{9}$$

Therefore in standard form we write

$$\frac{d^2\psi_1^-}{dx^2} - k^2\psi_1^- = 0, \tag{10}$$

where

$$k = \frac{\sqrt{-2mE}}{\hbar},$$

so that k is real and positive. The general solution to the Schrödinger Equation in region 1 is

$$\boxed{\psi_1^-(x) = Ae^{kx} + Be^{-kx}.} \tag{11}$$

In region 2 $(0 < x < a)$ the potential is finite: $V = -V_0$, so the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_2^-}{dx^2} = \psi_2^-(E + V_0), \tag{12}$$

which we can write as

$$\frac{d^2\psi_2^-}{dx^2} + q^2\psi_2^- = 0, \tag{13}$$

with

$$q = \frac{\sqrt{2m(E+V_0)}}{\hbar}.$$

The general solution in region 2 is

$$\boxed{\psi_2^-(x) = Ce^{iqx} + De^{-iqx}.} \tag{14}$$

By applying Bloch's Theorem as in equation (7), we can find the wavefunction belonging to negative Eigenvalues in the qth region of the Kronig-Penney model

$$\psi_q^-(x) = \begin{cases} \psi_1^-(x - p(a+b))e^{ip\mathrm{K}(a+b)}, & p(a+b) - b < x < p(a+b) \\ \psi_2^-(x - p(a+b))e^{ip\mathrm{K}(a+b)}, & p(a+b) < x < p(a+b) + a \end{cases} \tag{15}$$

where ψ_1^- is referenced for odd regions and ψ_2^- is referenced for even regions. Finally, by substituting in ψ_1^- and ψ_2^-, the wavefunction is

$$\psi_q^-(x) = \begin{cases} \left(Ae^{k(x-p(a+b))} + Be^{-k(x-p(a+b))}\right)e^{ip\mathrm{K}(a+b)}, & p(a+b) - b < x < p(a+b) \\ \left(Ce^{iq(x-p(a+b))} + De^{-iq(x-p(a+b))}\right)e^{ip\mathrm{K}(a+b)}, & p(a+b) < x < p(a+b) + a \end{cases} \tag{16}$$

where $p = \pm(0, 1, 2, \ldots)$, $k = \frac{\sqrt{-2mE}}{\hbar}$, $q = \frac{\sqrt{2m(E+V_0)}}{\hbar}$, $K = \frac{2\pi n}{N(a+b)}$. The integer p is the number of periods an arbitrary region is shifted with respect to the regions 1 or 2.

2.2: Applying Boundary Conditions to ψ^-:

Since we have four constants: $A, B, C,$ and D, we'll need four independent equations to normalize the wavefunctions. Two equations arise from the continuity requirement of the wavefunction at a boundary: the wavefunctions as well as their derivatives must be equal at any region boundaries. We only need to enforce boundary conditions at two locations. Therefore we can use the boundaries at $x = 0$ and $x = a$, and the wavefunctions from three regions: $\psi_1, \psi_2,$ and ψ_3.

The wavefunction in region 3, where $p = 1$, can be attained directly from equation (16)

$$\boxed{\psi_3^-(x) = \left(Ae^{k(x-a-b)} + Be^{-k(x-a-b)}\right)e^{i\mathrm{K}(a+b)}.} \tag{17}$$

We can apply boundary conditions, ensuring continuity of the wavefunction and its derivative. First at x=0, between regions (1) and (2):

$$\psi_1^-|_{x=0} = \psi_2^-|_{x=0} \Rightarrow A + B = C + D \tag{18}$$

$$\boxed{A + B - C - D = 0} \tag{19}$$

$$\left.\frac{d\psi_1^-}{dx}\right|_{x=0} = \left.\frac{d\psi_2^-}{dx}\right|_{x=0} \Rightarrow Ak - Bk = iCq - iDq \tag{20}$$

$$\boxed{Ak - Bk - iCq + iDq = 0} \tag{21}$$

Then at $x = a$, between regions (2) and (3):

$$\psi_2^-|_{x=a} = \psi_3^-|_{x=a} \Rightarrow Ce^{iqa} + De^{-iqa} = \left(Ae^{-kb} + Be^{kb}\right)e^{i\mathrm{K}(a+b)} \tag{22}$$

$$\boxed{\left(Ae^{-kb} + Be^{kb}\right)e^{i\mathrm{K}(a+b)} - Ce^{iqa} - De^{-iqa} = 0} \tag{23}$$

$$\left.\frac{d\psi_2^-}{dx}\right|_{x=a} = \left.\frac{d\psi_3^-}{dx}\right|_{x=a} \Rightarrow iCqe^{iqa} - iDqe^{-iqa} = \left(Ake^{-kb} - Bke^{kb}\right)e^{iK(a+b)} \tag{24}$$

$$\boxed{\left(Ake^{-kb} - Bke^{kb}\right)e^{iK(a+b)} - iCqe^{iqa} + iDqe^{-iqa} = 0} \tag{25}$$

Once we enforce continuity at the boundaries: $x = 0$ and $x = a$, Bloch's Theorem will ensure continuity at all other boundaries. Also, obtaining any more equations from other boundaries would be redundant, because the equations would simplify to one of the equations (19, 21, 23, 25).

2.3: Obtaining the Negative Eigenvalues of ψ^-:

We can assemble our system of equations (19, 21, 23, 25) into a matrix equation:

$$\mathbf{M}\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 & -1 \\ k & -k & -iq & iq \\ e^{-kb+i\mathrm{K}(a+b)} & e^{kb+i\mathrm{K}(a+b)} & -e^{iqa} & -e^{-iqa} \\ ke^{-kb+i\mathrm{K}(a+b)} & -ke^{kb+i\mathrm{K}(a+b)} & -iqe^{iqa} & iqe^{-iqa} \end{pmatrix}\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = 0 \tag{26}$$

This is a linear homogeneous system of equations. If the determinant of this matrix is not equal to zero, we know there is only the trivial solution, that all constants equal zero:

$$|\mathbf{M}| \neq 0 \;\Rightarrow\; A = B = C = D = 0 \tag{27}$$

However, if $|\mathbf{M}|$ is equal to zero, there are infinite solutions for $A, B, C,$ and D. From normalization we should be able to acquire a single solution. Before we can even attempt to find the constants: $A, B, C,$ and D, we need to find the Eigenvalues E, the energies for which possible states exists. So we'll assume the determinant is equal to zero:

$$|\mathbf{M}| = \begin{vmatrix} 1 & 1 & -1 & -1 \\ k & -k & -iq & iq \\ e^{-kb+i\mathrm{K}(a+b)} & e^{kb+i\mathrm{K}(a+b)} & -e^{iqa} & -e^{-iqa} \\ ke^{-kb+i\mathrm{K}(a+b)} & -ke^{kb+i\mathrm{K}(a+b)} & -iqe^{iqa} & iqe^{-iqa} \end{vmatrix} = 0 \tag{28}$$

After simplifying the determinant, and grouping like terms, we arrive at

$$\begin{aligned} 4ikq + e^{iK(a+b)}\big[&q^2\big(-e^{kb-iqa} + e^{kb+iqa} + e^{-kb-iqa} - e^{-kb+iqa}\big) \\ &+ ikq\big(-2e^{kb-iqa} - 2e^{kb+iqa} - 2e^{-kb-iqa} - 2e^{-kb+iqa}\big) \\ &+ k^2\big(e^{kb-iqa} - e^{-kb-iqa} - e^{kb+iqa} + e^{-kb+iqa}\big) + 4ikqe^{iK(a+b)}\big] = 0. \end{aligned} \tag{29}$$

Then we can combine the complex exponential pairs to form trigonometric functions:

$$\begin{aligned} 4ikq + e^{iK(a+b)}\big[&q^2\big(2ie^{kb}\sin(qa) - 2ie^{-kb}\sin(qa)\big) \\ &+ ikq\big(-4e^{kb}\cos(qa) - 4e^{-kb}\cos(qa)\big) \\ &+ k^2\big(-2ie^{kb}\sin(qa) + 2ie^{-kb}\sin(qa)\big) + 4ikqe^{iK(a+b)}\big] = 0. \end{aligned} \tag{30}$$

Also, we can convert the exponential pairs into hyperbolic trigonometric functions:

$$\begin{aligned} 4ikq = -4e^{iK(a+b)}[&q^2 i\sin(qa)\sinh(kb) - 2ikq\cos(qa)\cosh(kb) \\ &- k^2 i\sin(qa)\sinh(kb)] - 4ikqe^{2iK(a+b)} \end{aligned} \tag{31}$$

After some algebraic manipulation we have

$$\frac{1 + e^{2iK(a+b)}}{e^{iK(a+b)}} = \frac{(k^2 - q^2)}{kq}\sin(qa)\sinh(kb) + 2\cos(qa)\cosh(kb). \tag{32}$$

The left side can be converted into the complex exponential pair: $e^{-iK(a+b)} + e^{iK(a+b)}$, so

$$\boxed{\cos K(a+b) = \frac{k^2 - q^2}{2kq}\sin(qa)\sinh(kb) + \cos(qa)\cosh(kb).} \tag{33}$$

The solutions to this equation are the negative Eigenvalues of the Kronig-Penney model.

2.4: Numerically Solving for Bands and Gaps

The MATLAB script, kp_eig_plot_n.m, finds the negative Eigenvalues of the Kronig-Penney model. Then equation (33) is solved numerically using the MATLAB function true_intersect.m. All code is included in the appendix. The script produces the plot in figure 4 which depicts not only equation (33), but the

bands and forbidden gaps. To create this plot, the following settings were used: $V_0 = 1000\ eV,\ a = 1\text{Å},\ b = 0.1\ \text{Å},\ m = 9.109 \times 10^{-31}\ kg,\ N = 3,$ and $n = 3$.

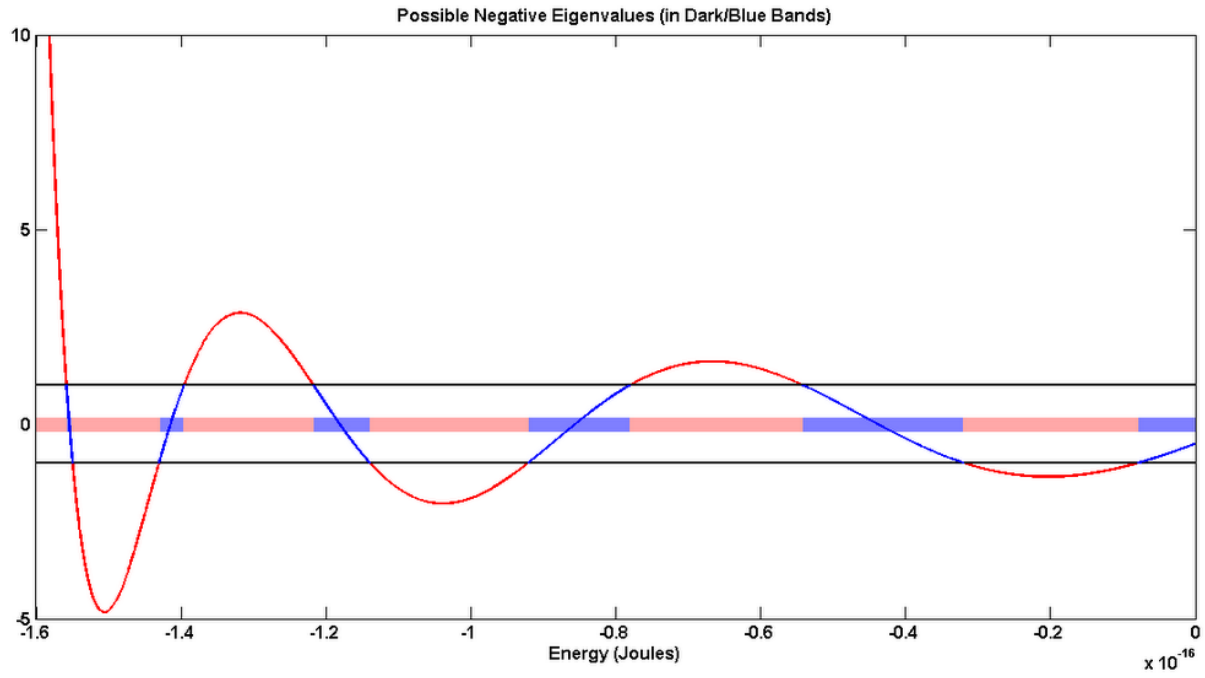

Figure 4: The Condition for Negative Eigenvalues

The right side of equation (33) is plotted in figure 4. Notice that on the left side of the equation (33), cosine can only provide a number between -1 and 1. These are upper and lower bounds, which are plotted in black. The curve in figure 4 is colored blue within these bounds, where possible Eigenvalues exist, and the bands are highlighted blue. The curve is colored red outside of the bounds and the corresponding forbidden gaps are highlighted pink. The curve never crosses into the bounds for energies less than the potential well depth, because the total energy of the particle can't be less than the potential well depth.

2.5 Normalization of ψ^-:

Typically normalization is completed such that the total probability of finding the particle anywhere is 1. Since we know the probability density function is periodic, just like the potential is periodic, we can normalize to the total probability of finding the particle in one period $\frac{1}{N}$:

$$\int_{-b}^{0} |\psi_1^-|^2 dx + \int_{0}^{a} |\psi_2^-|^2 dx = \frac{1}{N}. \tag{34}$$

If we substitute the wavefunctions from regions 1 and 2:

$$\int_{-b}^{0} \left|Ae^{kx} + Be^{-kx}\right|^2 dx + \int_{0}^{a} \left|Ce^{iqx} + De^{-iqx}\right|^2 dx = \frac{1}{N}. \tag{35}$$

2.6 Numerically Solving for Constants and Plotting Wavefunctions

The script kp_wave_n.m solves for the constants: $A, B, C, \text{and } D$. To start, $\mathbf{M}$ from equation (26), is saved as a symbolic matrix. Then all occurrences of E (and subsequently $k \text{ and } q$) are substituted with the Eigenvalue, creating a numerical matrix. Next the matrix is reduced into Row Reduced Echelon Form by Gauss-Jordan elimination, using the MATLAB function, rref(). We expect to obtain a rank 3 matrix, which leaves us with one constant which can be solved by normalization. Some complications were encountered in the normalization process, which are described in section 5.3. Therefore the wavefunctions are not normalized properly, but the shape of the probability density functions is correct.

Figure 5 shows a wavefunction provided by this method. Figure 5A is the real part of the wavefunction, 5B is the imaginary part, and 5C is the particle's location probability density function. To create these plots, the following settings were used: $V_0 = 1000\ eV,\ a = 1\ \text{Å},\ b = 1\ \text{Å},\ m = 9.109 \times 10^{-31}\ kg,$ $N = 10,\ \text{and } n = 1$. The fifth eigenvalue was selected.

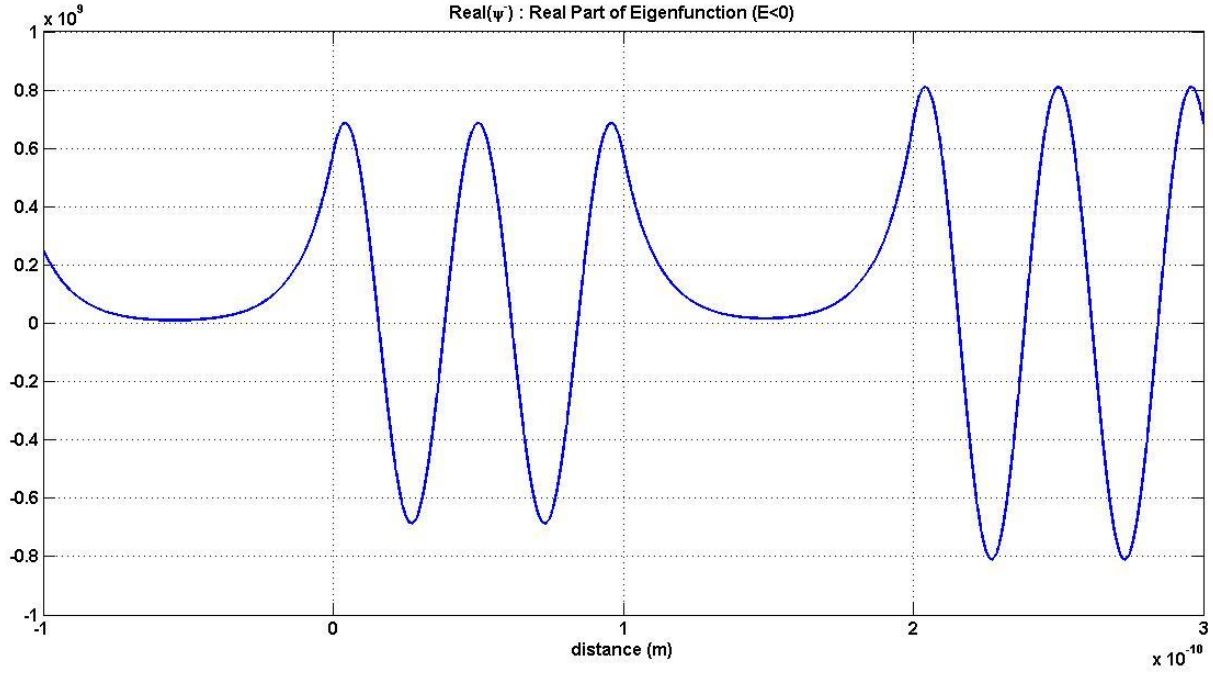

Figure 5A: Real Part of Eigenfunction for a Negative Eigenvalue

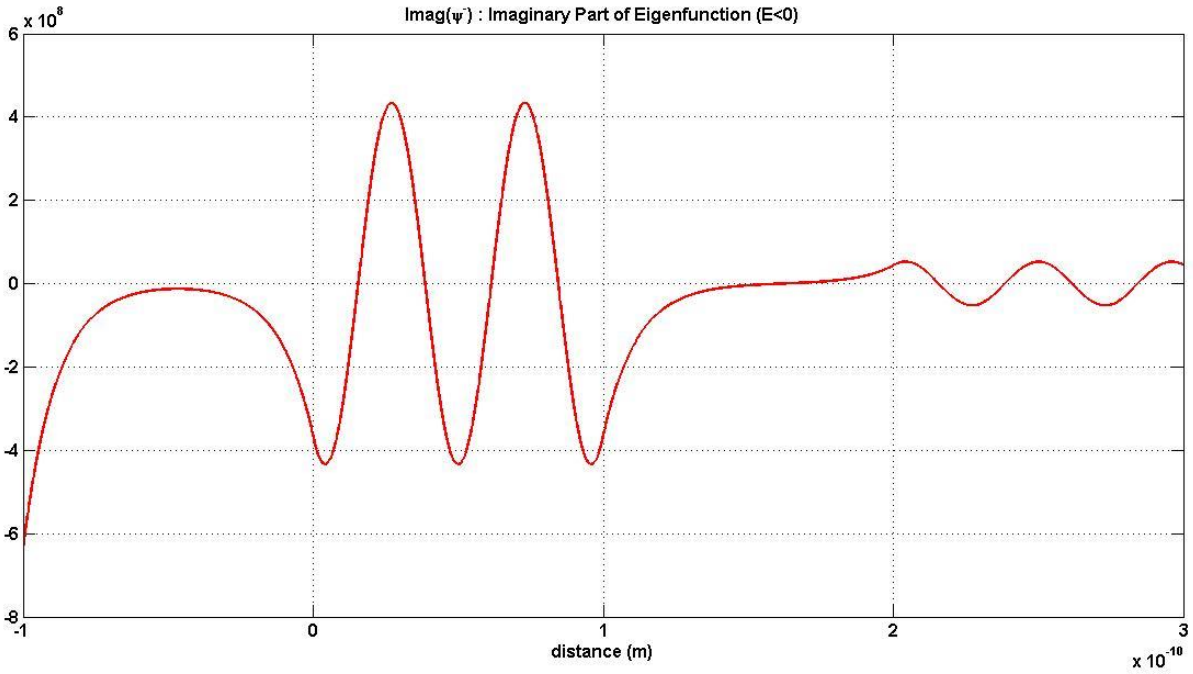

Figure 5B: Imaginary Part of Eigenfunction for a Negative Eigenvalue

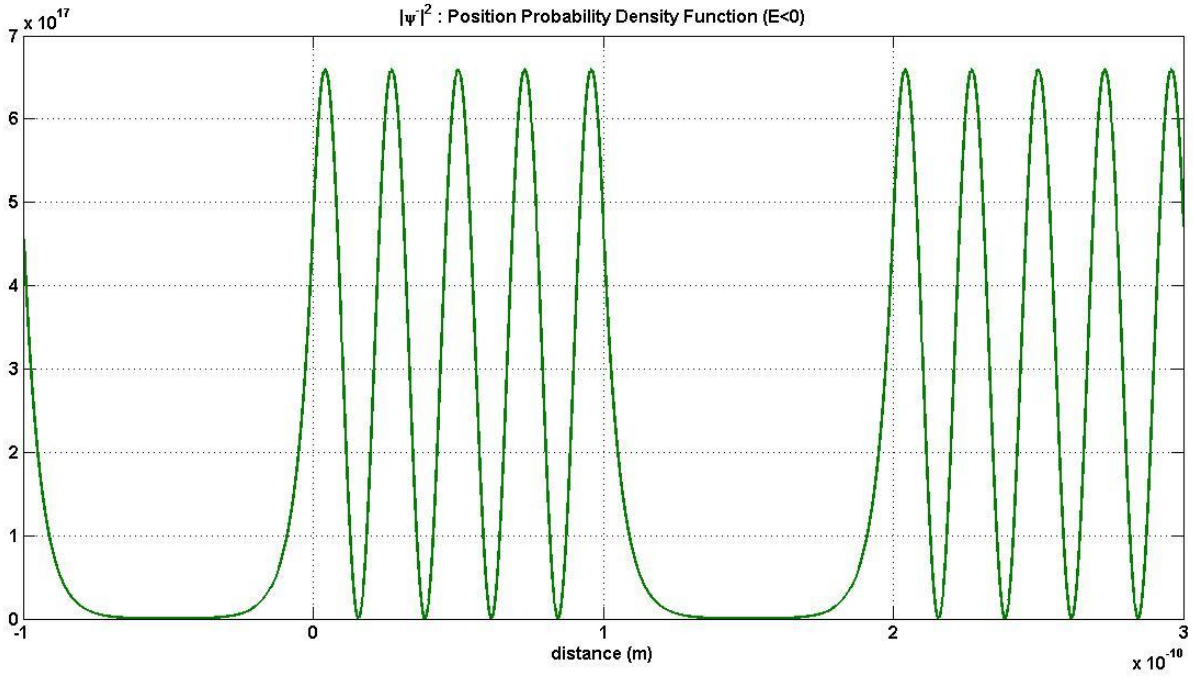

Figure 5C: Probability Density Function of Eigenstate for a Negative Eigenvalue

Part 3: Positive Total Energy: $E > 0$

In this section we seek the positive allowed energies of the Kronig-Penney model. First we'll solve the Schrödinger equation in three regions, as before, which will lead to the bands and gaps. Then we'll plot some possible wavefunctions.

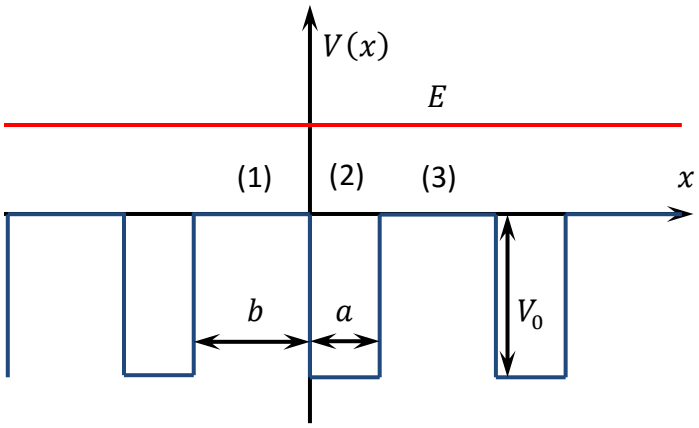

Figure 6: A Periodic Potential: $V(x)$ with Positive Total Energy: E

3.1: Finding Wavefunction within Regions 1 and 2 ($E > 0$): ψ_1^+, ψ_2^+

In region 1 $(-b < x < 0)$, where the potential is zero, the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_1^+}{dx^2} = E\psi_1^+. \tag{36}$$

Therefore in standard form we write

or,

$$\frac{d^2\psi_1^+}{dx^2} + u^2\psi_1^+ = 0, \quad \text{where } u = \frac{\sqrt{2mE}}{\hbar}, \tag{37}$$

so that u is real and positive.

The general solution in region 1, for positive Eigenvalues is

$$\boxed{\psi_1^+(x) = F\cos(ux) + G\sin(ux).} \tag{38}$$

In region 2 $(0 < x < a)$ the potential is finite: $V = -V_0$, so the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_2^+}{dx^2} = \psi_2^-(E + V_0). \tag{39}$$

In standard form we write:

$$\frac{d^2\psi_2^+}{dx^2} + w^2\psi_2^+ = 0, \quad \text{with } w = \frac{\sqrt{2m(E+V_0)}}{\hbar}. \tag{40}$$

The general solution in region 2, for positive Eigenvalues is

$$\boxed{\psi_2^+(x) = J\cos(wx) + L\sin(wx).} \tag{41}$$

From applying Bloch's Theorem in equations (7), the eigenfunctions belonging to the positive Eigenvalues in the qth region of the Kronig-Penney Model are

$$\psi_2^+(x) = \begin{cases} \big(F\cos\big(u(x-p(a+b)\big) + G\sin\big(u\,(x-p(a+b)\big)\,\big)e^{ip\mathrm{K}(a+b)}, & p(a+b)-b < x < p(a+b) \\ \big(J\cos\big(w(x-p(a+b)\big) + L\sin\big(w(x-p(a+b)\big)\big)e^{ip\mathrm{K}(a+b)}, & p(a+b) < x < p(a+b)+a \end{cases} \tag{42}$$

where $p = \pm(0,1,2,\dots)$, $u = \frac{\sqrt{2mE}}{\hbar}$, $w = \frac{\sqrt{2m(E+V_0)}}{\hbar}$, $K = \frac{2\pi n}{N(a+b)}$

Likewise, the eigenfunction in region 3 can be found by applying Bloch's Theorem:

$$\psi_3^+(x) = \psi_1^+(x-a-b)e^{i\mathrm{K}(a+b)}. \tag{43}$$

After substituting in ψ_1^+ we attain

$$\boxed{\psi_3^+(x) = \big(F\cos\big(u(x-a-b)\big) + G\sin\big(u(x-a-b)\big)\big)e^{i\mathrm{K}(a+b)}.} \tag{44}$$

3.2: Applying Boundary Conditions to ψ^+:

We can apply boundary conditions, ensuring continuity. First at x=0:

$$\psi_1^+|_{x=0} = \psi_2^+|_{x=0} \Rightarrow F = J \tag{45}$$

$$\boxed{F - J = 0} \tag{46}$$

$$\left.\frac{d\psi_1^+}{dx}\right|_{x=0} = \left.\frac{d\psi_2^+}{dx}\right|_{x=0} \Rightarrow -Fu(0) + Gu = -Jw(0) + Lw \tag{47}$$

$$\boxed{Gu - Lw = 0} \tag{48}$$

Then at x=a, between regions (2) and (3):

$$\psi_2^+|_{x=a} = \psi_3^+|_{x=a} \Rightarrow J\cos(wa) + L\sin(wa) = (F\cos(-ub) + G\sin(-ub))e^{iK(a+b)} \tag{49}$$

$$\boxed{(-F\cos(ub) + G\sin(ub))e^{iK(a+b)} + J\cos(wa) + L\sin(wa) = 0} \tag{50}$$

$$\left.\frac{d\psi_2^+}{dx}\right|_{x=a} = \left.\frac{d\psi_3^+}{dx}\right|_{x=a} \Rightarrow -Jw\sin(wa) + Lw\cos(wa) = (-Fu\sin(-ub) + Gu\cos(-ub))e^{iK(a+b)} \tag{51}$$

$$\boxed{(-Fu\sin(ub) - Gu\cos(ub))e^{iK(a+b)} - Jw\sin(wa) + Lw\cos(wa) = 0} \tag{52}$$

3.3: Finding Eigenvalues of ψ^+:

We can assemble our system of equations (46, 48, 50, 52) into a matrix equation:

$$\mathbf{Q}\begin{pmatrix} F \\ G \\ J \\ L \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & u & 0 & -w \\ -\cos(ub)\, e^{iK(a+b)} & \sin(ub)\, e^{iK(a+b)} & \cos(wa) & \sin(wa) \\ -u\sin(ub)\, e^{iK(a+b)} & -u\cos(ub)\, e^{iK(a+b)} & -w\sin(wa) & w\cos(wa) \end{pmatrix}\begin{pmatrix} F \\ G \\ J \\ L \end{pmatrix} = 0 \tag{53}$$

As before, we'll assume the determinant is equal to zero, to ensure the existence of nontrivial solutions of F, G, J, and L:

$$|\mathbf{Q}| = \begin{vmatrix} 1 & 0 & -1 & 0 \\ 0 & u & 0 & -w \\ -\cos(ub)\, e^{iK(a+b)} & \sin(ub)\, e^{iK(a+b)} & \cos(wa) & \sin(wa) \\ -u\sin(ub)\, e^{iK(a+b)} & -u\cos(ub)\, e^{iK(a+b)} & -w\sin(wa) & w\cos(wa) \end{vmatrix} = 0. \tag{54}$$

The determinant becomes

$$\begin{aligned} u[&w\cos^2(wa) + w\sin^2(wa) \\ &- w\left[-w\sin(ub)\sin(wa)\, e^{iK(a+b)} + u\cos(ub)\cos(wa)\, e^{iK(a+b)}\right] \\ &- u[w\left(\cos(ub)\cos(wa)\, e^{iK(a+b)}\right) - u\sin(ub)\sin(wa)\, e^{iK(a+b)}] \\ &+ w\left(u\cos^2(ub)\, e^{2iK(a+b)} + u\sin^2 ub\, e^{2iK(a+b)}\right) = 0. \end{aligned} \tag{55}$$

After some cancellation and collection of like-terms, we have

$$uw - e^{i\mathrm{K}(\mathrm{a}+\mathrm{b})}\left[-\sin(u\mathrm{b})\sin(w\mathrm{a})\,(u^2+w^2) + 2uw\cos(u\mathrm{b})\cos(w\mathrm{a}) - uw\,e^{i\mathrm{K}(\mathrm{a}+\mathrm{b})}\right] = 0. \tag{56}$$

Then we can collect the complex terms on the left-hand side:

$$\frac{uw\left(e^{2i\mathrm{K}(\mathrm{a}+\mathrm{b})}+1\right)}{e^{i\mathrm{K}(\mathrm{a}+\mathrm{b})}} = -\sin(u\mathrm{b})\sin(w\mathrm{a})\,(u^2+w^2) + 2uw\cos(u\mathrm{b})\cos(w\mathrm{a}), \tag{57}$$

which allows us to use Euler's identity to exchange them for a cosine function:

$$\boxed{\cos\left(\mathrm{K}(\mathrm{a}+\mathrm{b})\right) = -\frac{u^2+w^2}{2uw}\sin(u\mathrm{b})\sin(w\mathrm{a}) + \cos(u\mathrm{b})\cos(w\mathrm{a}).} \tag{58}$$

This equation provides us with the positive Eigenvalues which belong to the Hamiltonian of the state of the Kronig-Penney model.

3.4: Numerically Solving for Bands and Gaps

The MATLAB script kp_eig_plot_p.m, finds the positive Eigenvalues of the Kronig-Penney model. Then equation (58) is solved numerically. The following plot in figure 7 was produced, which depicts equation (58) along with the bands and forbidden gaps. To create this plot, the following settings were used: $V_0 = 1000\ eV,\ a = 1\ \text{Å},\ b = 0.1\ \text{Å},\ m = 9.109 \times 10^{-31}\ kg\ , N = 3,$ and $n = 3.$

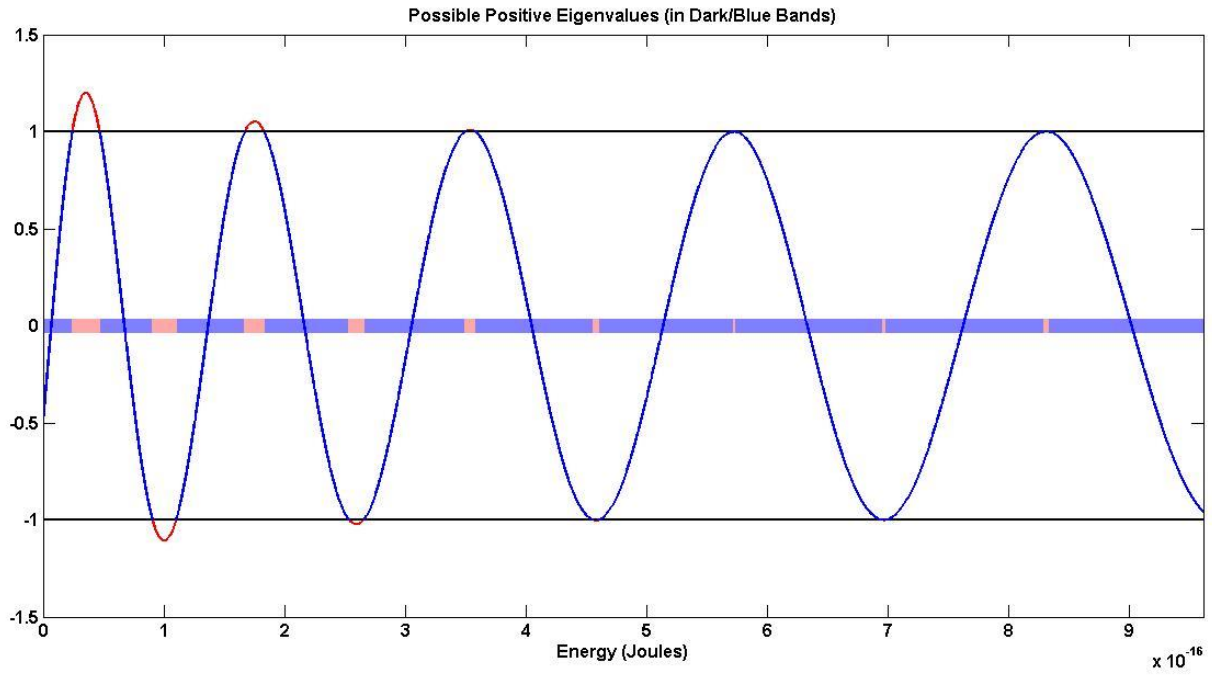

Figure 7: The Condition for Positive Eigenvalues

In figure 7 the right side of equation (58) is plotted. As before, on the left-hand side we notice that cosine can only provide a number between -1 and 1. Hence an upper and lower bound are plotted in black. The bands occur where the curve is within the bounds, shown as blue, whereas the forbidden gaps occur where the curve is outside the bounds and are shown in pink. As energy increases the bands become smaller and further apart.

3.5: Normalization of ψ^+:

Again, we can normalize to the total probability of finding the particle in one period $\frac{1}{N}$:

$$\int_{-b}^{0} |\psi_1^+|^2 dx + \int_{0}^{a} |\psi_2^+|^2 dx = \frac{1}{N}. \tag{59}$$

If we substitute the wavefunctions from regions 1 and 2:

$$\int_{-b}^{0} |F\cos(ux) + G\sin(ux)|^2 dx + \int_{0}^{a} |J\cos(wx) + L\sin(wx)|^2 dx = \frac{1}{N}. \tag{60}$$

3.6 Numerically Solving for Constants and Plotting Wavefunctions

The script kp_wave_p.m attempts to solves for the constants: F, G, J, and L, using the matrix, $\mathbf{Q}$ from equation (53). Complications which arose in the normalization process are described in section 5.3.

Figure 8 shows the numerically generated wavefunction. Figure 8A is the real part, 8B is the imaginary part, and 8C is the particle's location probability density function. To create these plots, the following settings were used: $V_0 = 1000\ eV$, $a = 1\ \text{Å}$, $b = 1\ \text{Å}$, $m = 9.109 \times 10^{-31}\ kg$, $N = 10$, and $n = 3$. The second eigenvalue was selected.

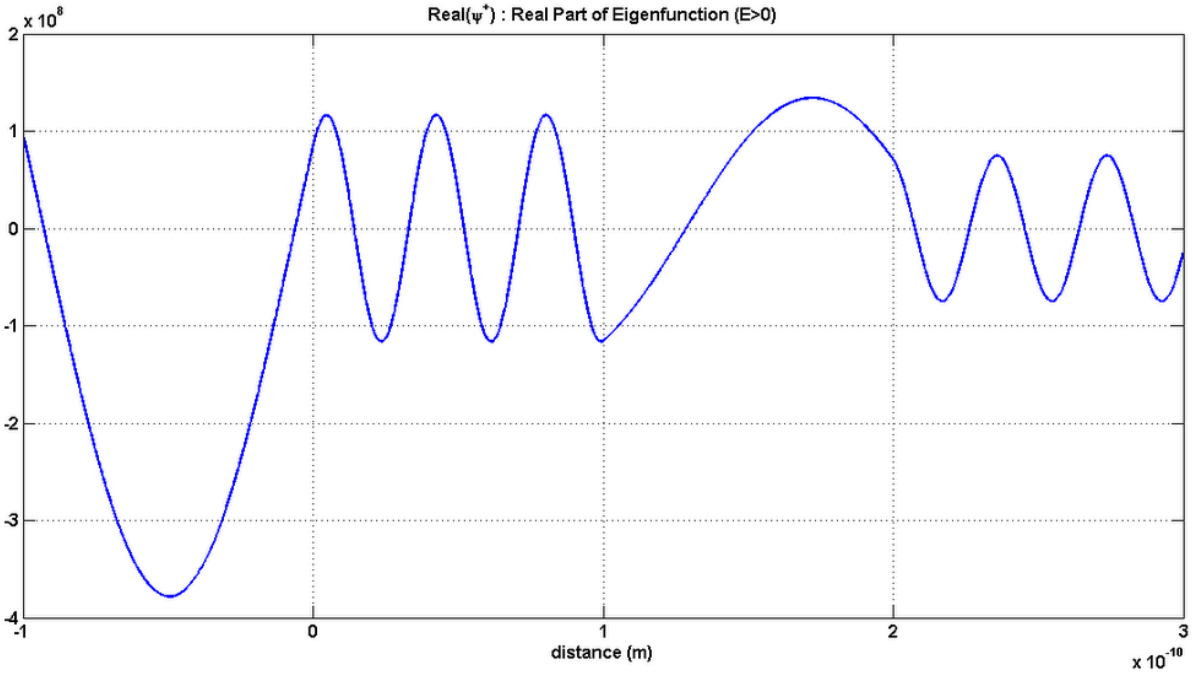

Figure 8A: Real Part of Eigenfunction for a Positive Eigenvalue

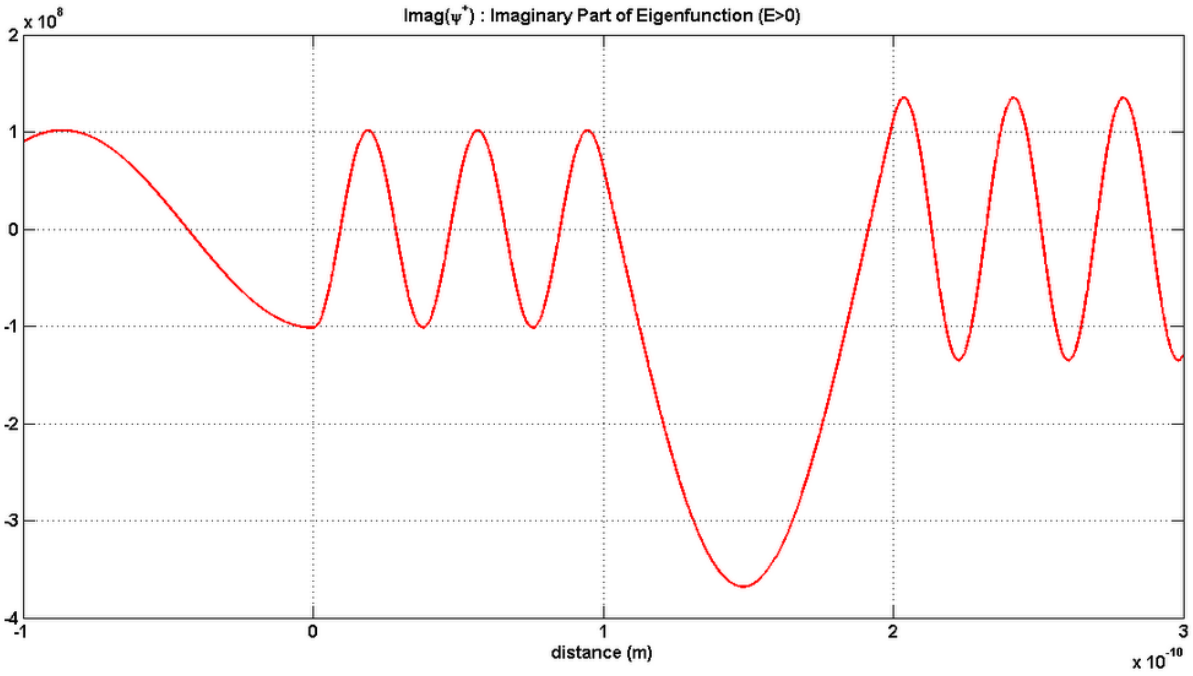

Figure 8B: Imaginary Part of Eigenfunction for a Positive Eigenvalue

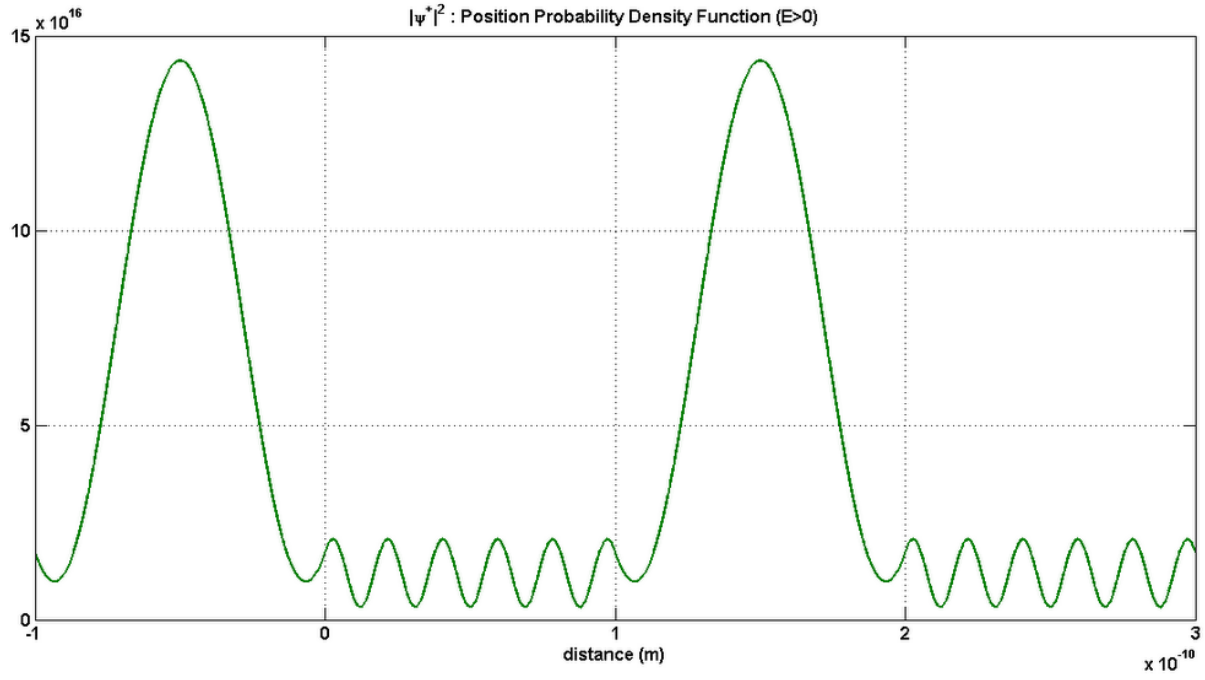

Figure 8C: Probability Density Function of Eigenstate for a Positive Eigenvalue

Part 4: Zero Total Energy: $E = 0$

In this section we're seeking the conditions for the eigenvalue, zero. Zero may be an eigenvalue if the dimensions of the system are just right. The dimensions are m, a, b, V_0, and N. Bands and gaps will be plotted, but this time it represents the allowed potential well depths: V_0, if the other dimensions are fixed. Also, some possible wavefunctions will be plotted.

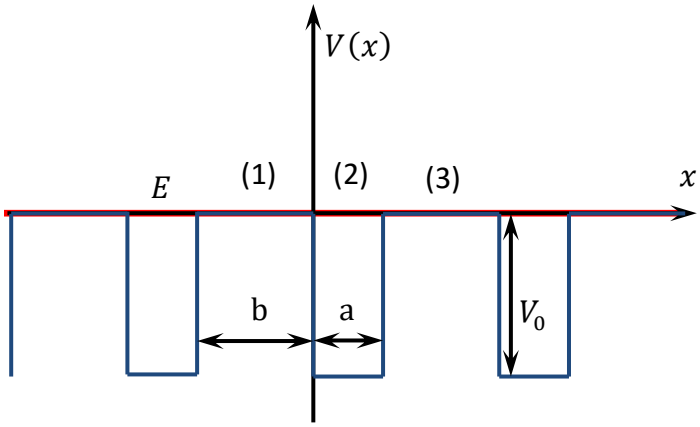

Figure 9: A Periodic Potential: $V(x)$ with Zero Total Energy: E

4.1: Finding Wavefunction within Regions 1 and 2: $\psi_1^=, \psi_2^=$

Both wavefunctions ψ^+ and ψ^- should be valid in their energy regions, but on their boundary at $E = 0$, we might find that they are inapplicable. Later we'll see whether the wavefunctions converge, or if $\psi^=$ is a unique wavefunction. For now, we can solve the Schrödinger equation for $E = 0$ to find $\psi^=$.

In region 1 $(-b < x < 0)$, where the potential is zero, the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_1^=}{dx^2} + (V_0 - E)\psi_1^= = 0, \tag{61}$$

and since $V_0 = E$ the equation becomes

$$\frac{d^2\psi_1^=}{dx^2} = 0, \tag{62}$$

Therefore we can integrate twice, and in doing so we introduce two constants, so the solution becomes

$$\psi_1^=(x) = \alpha x + \beta \tag{63}$$

In region 2 $(0 < x < a)$ the potential is finite: $V = -V_0$, so the Schrödinger equation becomes

$$-\frac{\hbar^2}{2m}\frac{d^2\psi_2^=}{dx^2} - V_0\psi_2^= = 0, \tag{64}$$

which we can write as

$$\frac{d^2\psi_2^=}{dx^2} + l^2\psi_2^= = 0,$$

with $$l = \frac{\sqrt{2mV_0}}{\hbar}. \tag{65}$$

The general solution in region 2 is

$$\psi_2^-(x) = \gamma\cos(lx) + \epsilon\sin(lx). \tag{66}$$

Then we can say the Eigenfunctions of the zero Eigenvalue in the Kronig Penney are

$$\boxed{\psi^-(x) = \begin{cases} (\alpha x + \beta\,)e^{ip\mathrm{K(a+b)}}, & p(\mathrm{a}+\mathrm{b}) - \mathrm{b} < x < p(\mathrm{a}+\mathrm{b}) \\ (\gamma\cos(lx) + \epsilon\sin(lx))e^{ip\mathrm{K(a+b)}}, & p(\mathrm{a}+\mathrm{b}) < x < p(\mathrm{a}+\mathrm{b}) + \mathrm{a} \end{cases}} \tag{67}$$

where $p = \pm(0, 1, 2, \ldots)$, $l = \frac{\sqrt{2mV_0}}{\hbar}$, $K = \frac{2\pi n}{N(\mathrm{a+b})}$. The integer p is the number of periods an arbitrary region is shifted with respect to the regions 1 or 2.

4.2: Applying Boundary Conditions to $\psi^=$:

The wavefunction, in region 3 can be found using Bloch's Theorem:

$$\psi_3^=(x) = \psi_1^=(x - \mathrm{a} - \mathrm{b})e^{i\mathrm{K(a+b)}}. \tag{68}$$

After substituting $\psi_1^=$ in we attain

$$\psi_3^=(x) = (\alpha(x - a - b) + \beta)e^{i\mathrm{K(a+b)}}. \tag{69}$$

We can apply some boundary conditions, ensuring continuity. First at $x = 0$:

$$\psi_1^=|_{x=0} = \psi_2^=|_{x=0} \Rightarrow [\alpha x + \beta]|_{x=0} = [\gamma\cos(lx) + \epsilon\sin(lx)]|_{x=0} \tag{70}$$

$$\boxed{\beta - \gamma = 0} \tag{71}$$

$$\left.\frac{d\psi_1^=}{dx}\right|_{x=0} = \left.\frac{d\psi_2^=}{dx}\right|_{x=0} \Rightarrow \alpha = [-\gamma l\sin(lx) + \epsilon l\cos(lx)]|_{x=0} \tag{72}$$

$$\boxed{\alpha - \epsilon l = 0} \tag{73}$$

Then at x=a, between regions (2) and (3):

$$\psi_2^=|_{x=\mathrm{a}} = \psi_3^=|_{x=\mathrm{a}} \Rightarrow [\gamma\cos(lx) + \epsilon\sin(lx)]|_{x=\mathrm{a}} = \left.(\alpha(x - a - b) + \beta)e^{i\mathrm{K(a+b)}}\right|_{x=\mathrm{a}} \tag{74}$$

$$\boxed{(-\alpha\mathrm{b} + \beta)e^{i\mathrm{K(a+b)}} - \gamma\cos(l\mathrm{a}) - \epsilon\sin(l\mathrm{a}) = 0} \tag{75}$$

$$\begin{aligned} \left.\frac{d\psi_2^-}{dx}\right|_{x=\mathrm{a}} = \left.\frac{d\psi_3^-}{dx}\right|_{x=\mathrm{a}} &\Rightarrow \left.\frac{d}{dx}[\gamma\cos(lx) + \epsilon\sin(lx)]\right|_{x=\mathrm{a}} \\ &= \left.\frac{d}{dx}\left[(\alpha(x - a - b) + \beta)e^{i\mathrm{K(a+b)}}\right]\right|_{x=\mathrm{a}} \end{aligned} \tag{76}$$

$$\boxed{\alpha e^{i\mathrm{K(a+b)}} + \gamma l \sin(l\mathrm{a}) - \epsilon l \cos(l\mathrm{a}) = 0} \tag{77}$$

4.3: Conditions for the Existence of the Zero Eigenstate:

We already have a particular Eigenstate in mind here, so now our search turns to conditions that give rise to the zero Eigenstate, the dimensions: a, b, m, V_0 and N.

Again we assemble our system of equations into a matrix equation:

$$\mathbf{R}\begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \epsilon\end{pmatrix} = 0$$

$$= \begin{pmatrix} 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -l \\ -be^{i\mathrm{K(a+b)}} & e^{i\mathrm{K(a+b)}} & -\cos(\mathrm{a}l) & -\sin(\mathrm{a}l) \\ e^{i\mathrm{K(a+b)}} & 0 & l\sin(\mathrm{a}l) & -l\cos(\mathrm{a}l) \end{pmatrix}\begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \epsilon\end{pmatrix} = 0 \tag{78}$$

As before, we'll assume the determinant is equal to zero, to ensure the existence of nontrivial solutions of α, β, γ, and ϵ:

$$|\mathbf{R}| = \begin{vmatrix} 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -l \\ -be^{i\mathrm{K(a+b)}} & e^{i\mathrm{K(a+b)}} & -\cos(\mathrm{a}l) & -\sin(\mathrm{a}l) \\ e^{i\mathrm{K(a+b)}} & 0 & l\sin(\mathrm{a}l) & -l\cos(\mathrm{a}l) \end{vmatrix} = 0. \tag{79}$$

The determinant becomes

$$\begin{aligned} -\left[l\cos^2(\mathrm{a}l) + l\sin^2(\mathrm{a}l) - l\left(-\mathrm{b}l\sin(\mathrm{a}l)\, e^{i\mathrm{K(a+b)}} + \cos(\mathrm{a}l)\, e^{i\mathrm{K(a+b)}}\right)\right] \\ -\left[-l\cos(\mathrm{a}l)\, e^{i\mathrm{K(a+b)}} - l\left(-e^{2i\mathrm{K(a+b)}}\right)\right] = 0. \end{aligned} \tag{80}$$

After cancelling a couple terms and factoring out some others, we have

$$-l + e^{i\mathrm{K(a+b)}}\left[-\mathrm{b}l^2\sin(\mathrm{a}l) + 2l\cos(\mathrm{a}l) - le^{i\mathrm{K(a+b)}}\right]. \tag{81}$$

Then we can collect the complex terms on the left-hand side:

$$\frac{1 + e^{2i\mathrm{K(a+b)}}}{e^{i\mathrm{K(a+b)}}} = -\mathrm{b}l\sin(\mathrm{a}l) + 2\cos(\mathrm{a}l), \tag{82}$$

As in the previous cases we can make the left-hand side real, using Euler's identity:

$$\boxed{\cos\big(\mathrm{K(a+b)}\big) = -\frac{\mathrm{b}l}{2}\sin(\mathrm{a}l) + \cos(\mathrm{a}l).} \tag{83}$$

This equation is the condition for the zero Eigenstate, which is ultimately a function of $V_0, \mathrm{a}, \mathrm{b}, n, \text{and } N$.

4.6: Numerically Solving for Bands and Gaps

The same format was used as before in writing the MATLAB script, kp_eig_plot_eq.m, to find the potential well depths V_0 which allow for an Eigenvalues of zero. Constants such as a, b, $m, N,$ and n are predefined. Then equation (83) is solved numerically. The following plot was produced which depicts not only equation (83), but the bands and forbidden gaps. To create this plot, the following settings were used: $a = 1\ \text{Å}, b = 0.1\ \text{Å}, m = 9.109 \times 10^{-31}\ kg,\ N = 10,\ \text{and } n = 1.$

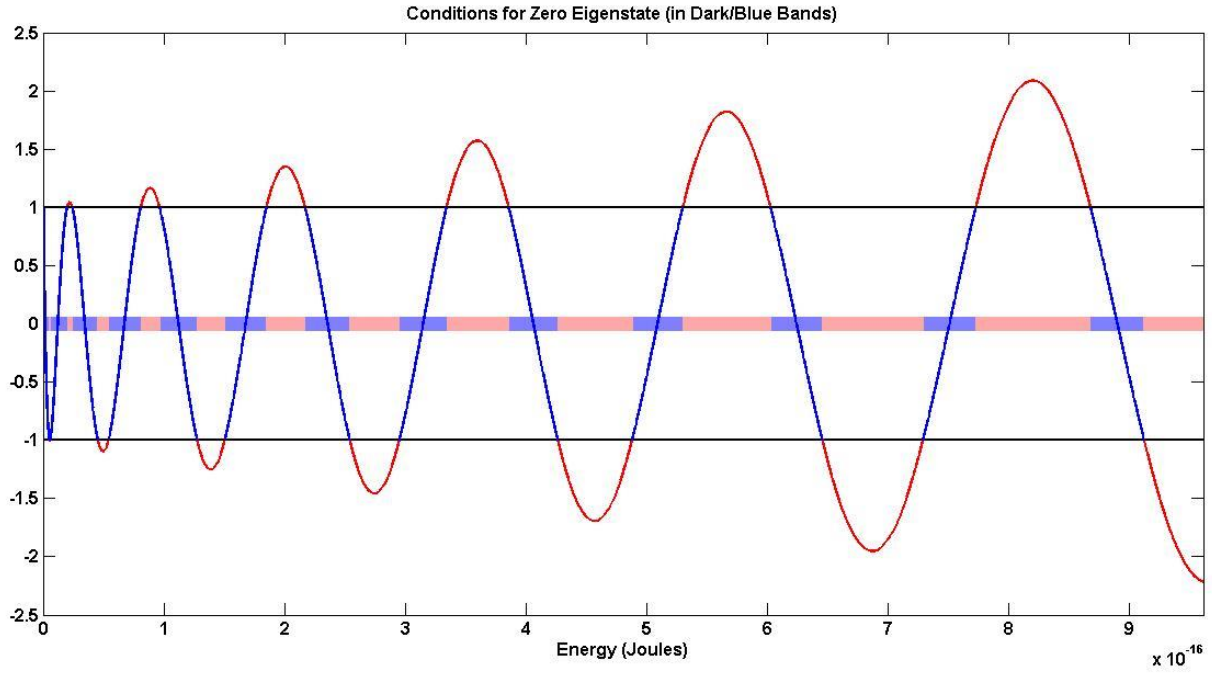

Figure 10: The Condition for an Eigenvalue of Zero

The right side of equation (83) is plotted in figure 10. Again, the upper and lower bounds of the left side of the equation ± 1 are plotted as black horizontal lines. The curve in figure 10 is colored blue within these bounds, where possible well depths exist, and the bands are highlighted blue. The curve is colored red outside of the bounds and the corresponding forbidden gaps are highlighted pink.

4.5: Normalization of $\psi^{=}$:

Typically normalization is done, such that the total probability of finding the particle anywhere is 1. Since we know the probability density function is periodic, just like the potential is periodic, we can normalize to the total probability of finding the particle in one period $\frac{1}{N}$:

$$\int_{-b}^{0} |\psi_1^{=}|^2 dx + \int_{0}^{a} |\psi_2^{=}|^2 dx = \frac{1}{N}. \tag{84}$$

If we substitute the wavefunctions from regions 1 and 2:

$$\int_{-b}^{0} |\alpha x + \beta|^2 dx + \int_{0}^{a} |\gamma \cos(lx) + \epsilon \sin(lx)|^2 dx = \frac{1}{N}. \tag{85}$$

4.6 Simulation: Numerical Solution of Normalized Constants

The script kp_wave_eq.m attempts to solves for the constants: $\alpha, \beta, \gamma, \text{and } \epsilon$, using the matrix, $\mathbf{R}$ from equation (78). Complications which arose in the normalization process are described in section 5.3.

In figure 11A, the real part of the wavefunctions is shown, in figure 11B the imaginary part is shown, and in figure 11C, the probability density function is shown. To create these plots, the following settings were used: $V_0 = 169.2\, eV,\ a = 1\,\text{Å},\ b = 1\,\text{Å},\ m = 9.109 \times 10^{-31}\, kg,\ N = 10,\ \text{and } n = 3$. For each plot in figure 11, a sub-plot is shown using the wavefunction for E>0, E<0 and E=0. A potential well depth was chosen such that the wavefunction corresponding to E=0 could be plotted using all three equations, (16), (42) and (67). For some reason the E<0 plot does not match the others. The numerical tolerance in the (reduced row echelon form) rref() MATLAB function was reduced to get better accuracy when solving for $\mathbf{M}$. Unfortunately this resulted in the trivial solution in which all constants are equal to zero. Notice that in figure 11C, the probability functions have the same shape, but the first (E<0) plot has a greater scale.

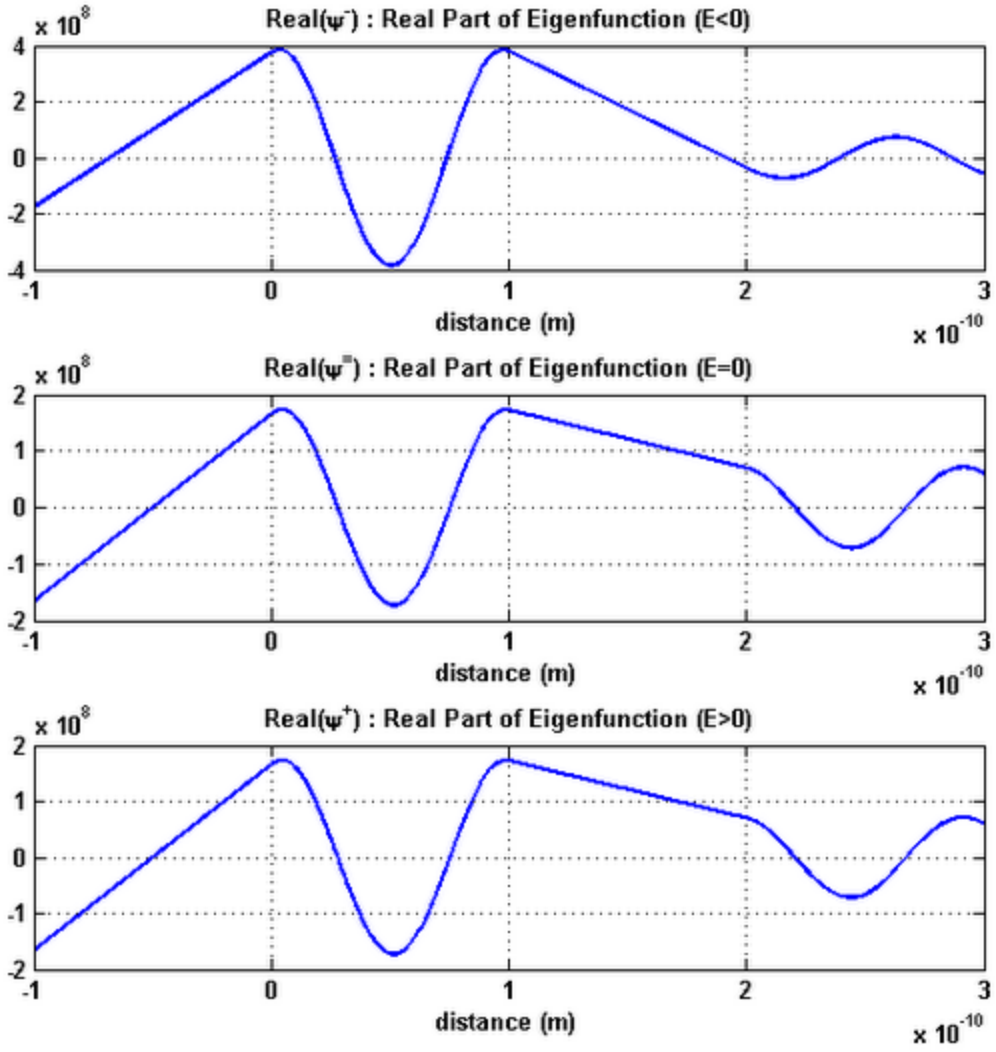

Figure 11A: Real Part of Eigenfunction for an Eigenvalue of Zero

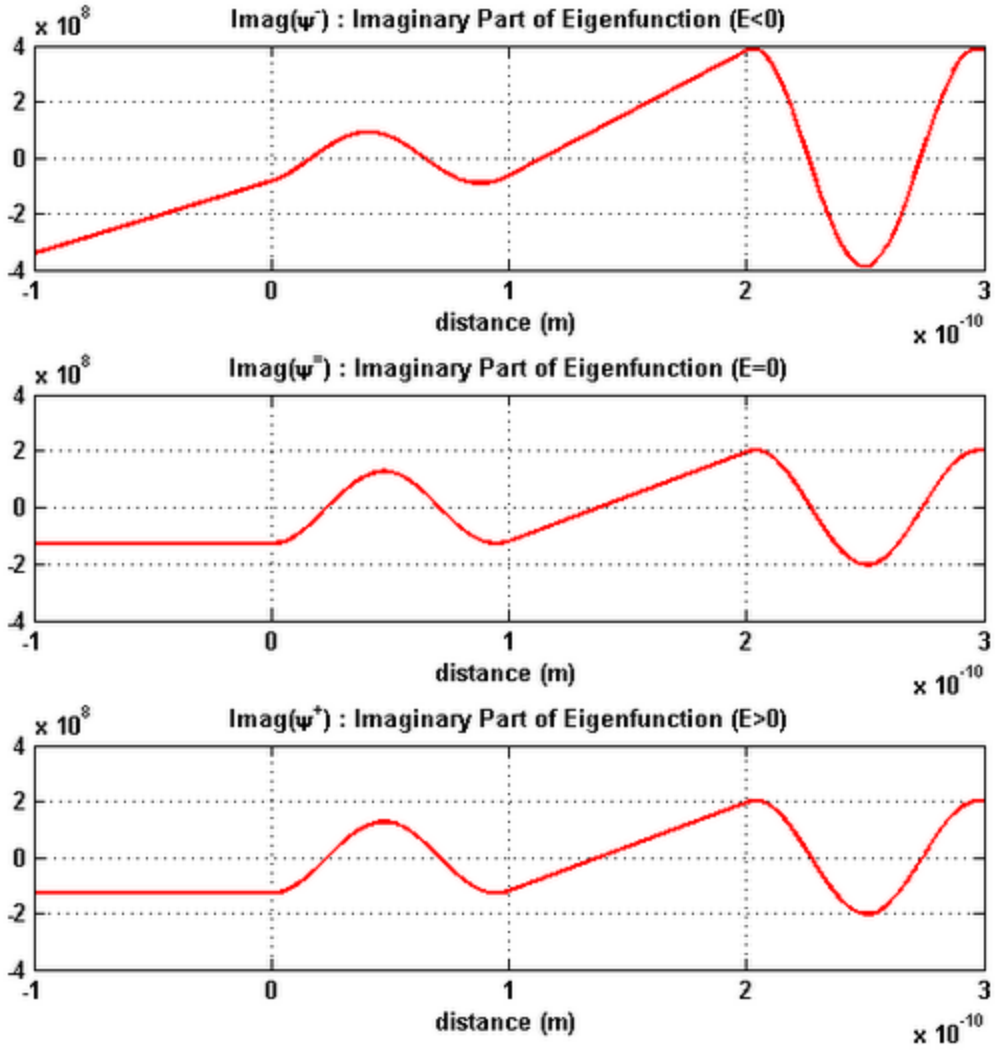

Figure 11B: Real Part of Eigenfunction for an Eigenvalue of Zero

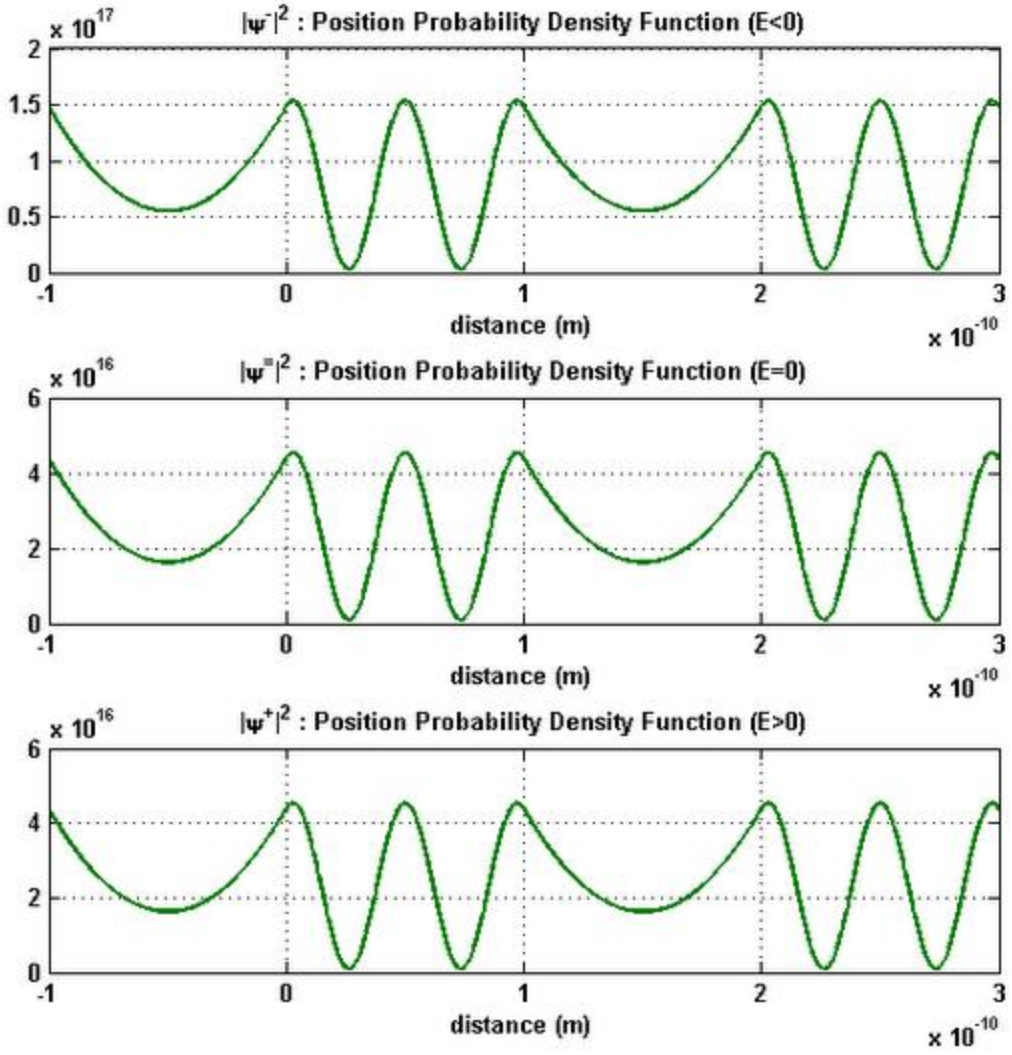

Figure 11C: Real Part of Eigenfunction for an Eigenvalue of Zero

Part 5: Discussion

5.1: Comparing the Eigenvalue Equations:

For each energy case: E<0, E=0, and E>0, we found a different equation that provides us with the Eigenvalues. By now you may have guessed that equations (33) and (58) converge to the E=0 case, equation (83), as $E \to 0$. We'll examine the Eigenvalue equations for |E|>0 at the boundary where E=0. The left side of each equation is the same: $\cos\big(\mathrm{K(a+b)}\big)$, so we'll focus on the right sides.

For E<0 we can take the limit of the right side of equation (33) as $E \to 0$. More directly, we can take the limit as $k \to 0$ and $q \to l$, while using L'Hopital's rule, to find the right side of equation (83):

$$\lim_{k\to 0}\left(\frac{k^2-l^2}{2kl}\sin(al)\sinh(bl)+\cos(al)\cosh(bl)\right)=-\frac{\mathrm{b}l}{2}\sin(\mathrm{a}l)+\cos(\mathrm{a}l)\,. \tag{86}$$

Likewise for E>0, the limit of the right side of equation (62) as $u \to 0$ and $w \to l$, equals the right side of equation (90):

$$\lim_{u\to 0}\left(-\frac{u^2+w^2}{2uw}\sin(u\mathrm{b})\sin(w\mathrm{a})+\cos(u\mathrm{b})\cos(w\mathrm{a})\right)=-\frac{\mathrm{b}l}{2}\sin(\mathrm{a}l)+\cos(\mathrm{a}l) \tag{87}$$

The Eigenvalue equation for E=0 is simply a special case of either Eigenvalue equation for |E|>0. This revelation might lead you to question whether these Eigenvalue equations are equivalent, such that a single equation describes the system at all possible energies.

To investigate if the equations are equivalent, we can start with equation (58) for E>0, but allow E to be negative and see if the result is equivalent to equation (33). Equation (83) is repeated here:

$$\cos\big(\mathrm{K(a+b)}\big)=-\frac{u^2+w^2}{2uw}\sin(u\mathrm{b})\sin(w\mathrm{a})+\cos(u\mathrm{b})\cos(w\mathrm{a})$$

By making E negative and replacing w with q, we can compare the equations, using the following relationships:

- u is imaginary, so the denominator becomes imaginary: $2uw \leftrightarrow i(2kq)$.
- u^2 is negative, so $-u^2-w^2 \leftrightarrow k^2-q^2$
- $\sin(ub)=\dfrac{e^{iub}-e^{-iub}}{2i} \leftrightarrow \dfrac{-1}{i}\dfrac{e^{kb}-e^{-kb}}{2}=\dfrac{-1}{i}\sinh(kb)$
- $\cos(ub)=\dfrac{e^{iub}+e^{-iub}}{2} \leftrightarrow \dfrac{e^{kb}+e^{-kb}}{2}=\cosh(kb)$

Therefore, equation (62) becomes:

$$\cos\big(\mathrm{K(a+b)}\big)=\frac{k^2-q^2}{i(2kq)}\left(\frac{-1}{i}\right)\sinh(\mathrm{kb})\sin(q\mathrm{a})+\cosh(k\mathrm{b})\cos(q\mathrm{a}), \tag{88}$$

which can be simplified to equation (33). Therefore equations (33) and (58) carry the same information. Either of them are adequate to provide all Eigenvalues of the system. Also, equation (83) is contained in either equation.

5.2: Complications in Simulation

In sections 2.6, 3.6, and 4.6 I mentioned that there were complications in numerically solving for the constants A, B, C, D, etc. After reducing the matrices $\mathbf{M}$, $\mathbf{Q}$, or $\mathbf{R}$, I expected to find a rank 3 matrix. In Reduced-Row Echelon form every row must have its' leading term at least one element to the right of the previous row's leading term. In a 4x4 matrix, this would require that the last row must be: "0 0 0 1" or "0 0 0 0". The former case implies that D is 0. The latter implies that D is a free term. Hence we have the trivial solution that $D = 0$ and infinite nontrivial solutions, respectively. Homogeneous systems always have infinite nontrivial solutions. Then after normalizing, we should have a single solution.

Depending on the selected parameters, the reduced row matrix was rank 1,2,3 or 4. It seems that numerical tolerances used by the software caused it to decide that more than two of the rows of the reduced matrix were linearly dependent. Occasionally the reduced matrix was rank 4, the identity matrix, which results in the trivial solution where all constants are zero. The rref() function has a tolerance than can be decreased, which sometimes deters the software from imposing a false linearity on the resulting equations. Unfortunately, decreasing this tolerance sometimes caused the software to find the trivial solution, which I pointed out in section 4.6.

In section 2, for the cases with rank 2 or 3, the equations (19, 21, 23, 25) were solved symbolically, after substituting the results from the reduced matrix. Ideally this would be redundant, but it was needed here because of the apparent low rank. So in the case of rank 2, one of these equations was solved, and for rank 1, two of them were solved. Control flow was used to handle these various cases. Similar procedures were used in sections 3 and 4.

Finally, while the normalization method from section 2.5, 3.5, and 4.5 are correct in principle, they could not be achieved numerically. For example, when attempting to manipulate equation (35) in section 2.5, to solve for D, we arrive at

$$\int_{-b}^{0} \left| \frac{A}{D} e^{kx} + \frac{B}{D} e^{-kx} \right|^2 dx + \int_{0}^{a} \left| \frac{C}{D} e^{iqx} + e^{-iqx} \right|^2 dx = \frac{1}{N|D|^2}. \tag{89}$$

In solving for $\mathbf{M}$ we attain $\frac{A}{D}, \frac{B}{D}, \text{and } \frac{C}{D}$, so the left side of equation (89) can be numerically integrated easily. But notice that we can only solve for $|D|$. All of the wavefunction plots that I've shown assume that D is real and positive, which is not a general requirement of the system. It appears that the system has a phase ambiguity. If somehow the phase of D could be found then we could easily find the phase of the remaining constants. However we can be confident that the probability density functions are accurate because they are independent of the phase of D. It's important to remember that these are

wavefunctions of a model and they're only an approximation of the true wavefunctions within the one-dimensional lattice.

5.3: Conclusion

You have seen the Kronig-Penney model used to approximate the potential energy function of a one-dimensional crystal lattice. This square well approximation allows us to find close approximations of the eigenvalues of a particle in a one-dimensional crystal lattice. By applying an eigenvalue to the general wavefunctions, their constants can be solved for. Figure 12 shows eigenvalues from within and outside of the well. Notice that within the well there are only a few narrow bands. But if the particle happens to be freed from the well there are much wider bands with only narrow forbidden gaps.

I encountered difficulties in the process of normalization, so one of the constants in the wavefunctions was always assumed to be real and positive. The true phase of this constant, might be solvable, or it could be inherently ambiguous. Examples of wavefunctions were shown, assuming that one of the constants was real and positive, which might not be true in general. However, the resulting probability density functions are not dependent on this phase, so they are truly representative of the Kronig-Penney Model.

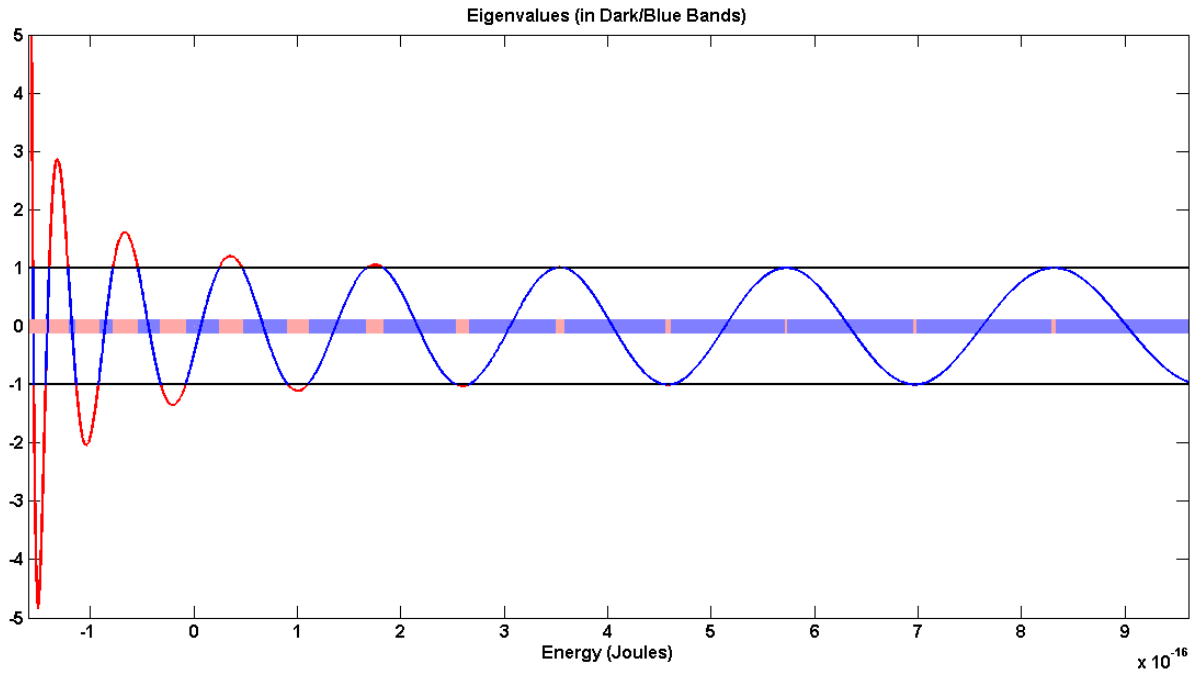

Figure 7: Bands and Forbidden Gaps

Appendix:

kp_eig_plot_n.m:

```
% kp_eig_plot_n.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Negative Eigenvalue Bands

clc; clear;

V0=1000;         % Potential Depth (eV)
a = 1e-10;       % Lengths a and b (m)
b = 1e-11;

m = 9.109e-31;   % Mass of electron (kg)
N = 3;           % Number of periods in crystal
n = 3;           % Propagation constant, determines phase of wavefunction
h = 6.626e-34 / (2*pi); % Planck's Constant:h_bar (J*s)
J_eV = 1.602e-19;       % J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

% Symbolic Variables, all simplified in terms of E:
syms E k q;
k = ((-2*m*E)^0.5)/h;
q = (2*m*(E+V0J))^0.5/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);

% Equation 33, the condition for negative Eigenvalues.
rhs = ( (k^2-q^2)/(2*k*q))*sinh(k*b)*sin(q*a)+cosh(k*b)*cos(q*a);
lhs=cos(Kab);

% Generate potential energy indices and substitute into rhs.
pLen=100000;
begin = -V0J
finish = 0
En=linspace(begin,finish,pLen);
RHS=subs(rhs,E,En);

% Plot dimensions, x-y limits:
figure1 = figure;
axes1 = axes('Parent',figure1);


% Plot entire curve in red. Certain parts will be covered in blue.
plot1(1)=plot(En,RHS, 'r');

% Generate upper and lower bounds.
upp=ones(1,pLen);
low=-1*ones(1,pLen);
```

```
% Generate bands and gaps for plot.
Eigval=RHS.*(RHS<1).*(RHS>-1);
Eigval(isnan(Eigval))=0;
gaps=Eigval;
gaps(Eigval~=0)=1;
gaps=not(gaps);
Eigval(Eigval==0)=NaN;
bands=Eigval*0;
gaps=double(gaps);
gaps(gaps==0)=NaN;
gaps = gaps*0;

% Plot bounds, bands, gaps, and portion of curve between bounds.
hold on
plot1(2)=plot(En,upp,'k');
plot1(3)=plot(En,low,'k');
plot1(4)=plot(En,bands);
plot1(6) = plot(En,gaps);
plot1(5)=plot(En,Eigval,'b');

% Set line widths and use custom colors.
set(plot1(1),'LineWidth',2,'Color',[1 0 0]);
set(plot1(2),'LineWidth',2,'Color',[0 0 0]);
set(plot1(3),'LineWidth',2,'Color',[0 0 0]);
set(plot1(5),'LineWidth',2,'Color',[0 0 1]);
set(plot1(4),'LineWidth',10,'Color',[.5 .5 1]);
set(plot1(6),'LineWidth',10,'Color',[1 .66 .66]);

% Define labels and axes size.
xlim([begin finish])
ylim([-5 10])
set(gca,'FontSize',12,'fontWeight','bold')
title('Possible Negative Eigenvalues (in Dark/Blue Bands)')
xlabel('Energy (Joules)')
```

kp_eig_plot_p.m:

```
% kp_eig_plot_p.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Positive Eigenvalue Bands

clc; clear;

V0=1000;          %Potential Depth (eV)
a = 1e-10;           %Width a & b (m)
b = 1e-11;

m = 9.109e-31;       %Mass of electron (kg)
N = 3;            % Number of periods in crystal
```

```
n = 3;              %Propagation constant, determines phase of wavefunction
h=6.626e-34 / (2*pi); %Planck's Constant:h_bar) (J*s)
J_eV = 1.602e-19;     %J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

%Symbolic Variables, all simplified in terms of E
syms E k q;
u = ((2*m*E)^0.5)/h;
w = (2*m*(E+V0J))^0.5/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);

% Equation 58, the condition for positive Eigenvalues.
rhs = ( -(u^2+w^2)/(2*u*w))*sin(u*b)*sin(w*a)+cos(u*b)*cos(w*a);
lhs=cos(Kab);

% Generate potential energy indices and substitute into rhs.
pLen=100000;
begin = 0
finish = 6*V0J
En=linspace(begin,finish,pLen);
RHS=subs(rhs,E,En);

% Plot dimensions, x-y limits:
figure1 = figure;
axes1 = axes('Parent',figure1);


% Plot entire curve in red. Certain parts will be covered in blue.
plot1(1)=plot(En,RHS, 'r');

% Generate upper and lower bounds.
upp=ones(1,pLen);
low=-1*ones(1,pLen);

% Generate bands and gaps for plot.
Eigval=RHS.*(RHS<1).*(RHS>-1);
Eigval(isnan(Eigval))=0;
gaps=Eigval;
gaps(Eigval~=0)=1;
gaps=not(gaps);
Eigval(Eigval==0)=NaN;
bands=Eigval*0;
gaps=double(gaps);
gaps(gaps==0)=NaN;
gaps = gaps*0;

% Plot bounds, bands, gaps, and portion of curve between bounds.
hold on
plot1(2)=plot(En,upp,'k');
plot1(3)=plot(En,low,'k');
plot1(4)=plot(En,bands);
plot1(6) = plot(En,gaps);
```

```
plot1(5)=plot(En,Eigval,'b');

% Set line widths and use custom colors.
set(plot1(1),'LineWidth',2,'Color',[1 0 0]);
set(plot1(2),'LineWidth',2,'Color',[0 0 0]);
set(plot1(3),'LineWidth',2,'Color',[0 0 0]);
set(plot1(5),'LineWidth',2,'Color',[0 0 1]);
set(plot1(4),'LineWidth',10,'Color',[.5 .5 1]);
set(plot1(6),'LineWidth',10,'Color',[1 .66 .66]);

% Define labels and axes size.
xlim([begin finish])
set(gca,'FontSize',12,'fontWeight','bold')
title('Possible Positive Eigenvalues (in Dark/Blue Bands)')
xlabel('Energy (Joules)')
```

kp_eig_plot_eq.m:

```
% kp_eig_plot_eq.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Energy Condition for Zero-Eigenvalue

clc; clear;

V0=1000;          % Reference Potential Depth (eV)
a = 1e-10;          %Width a & b (m)
b = 1e-11;

m = 9.109e-31;      %Mass of electron (kg)
N = 10;            % Number of periods in crystal
n = 1;             %Propagation constant, determines phase of wavefunction
h=6.626e-34 / (2*pi); %Planck's Constant:h_bar) (J*s)
J_eV = 1.602e-19;     %J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

%Symbolic Variables, all simplified in terms of E:
syms E l;
l = ((2*m*E)^0.5)/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);


% Equation 83, the condition for an Eigenvalue of 0.
rhs = ( cos(l*a)-(l*b/2)*sin(l*a));
lhs=cos(Kab);

pLen=100000;
begin = 0
```

```
finish = 6*V0J
En=linspace(begin, finish,pLen);
RHS=subs(rhs,E,En);

%Plot dimensions, x-y limits
figure1 = figure;
axes1 = axes('Parent',figure1);

% Plot entire curve in red. Certain parts will be covered in blue.
plot1(1)=plot(En,RHS, 'r');

% Generate upper and lower bounds.
upp=ones(1,pLen);
low=-1*ones(1,pLen);



% Generate bands and gaps for plot.
Eigval=RHS.*(RHS<1).*(RHS>-1);
Eigval(isnan(Eigval))=0;
gaps=Eigval;
gaps(Eigval~=0)=1;
gaps=not(gaps);
Eigval(Eigval==0)=NaN;
bands=Eigval*0;
gaps=double(gaps);
gaps(gaps==0)=NaN;
gaps = gaps*0;

% Plot bounds, bands, gaps, and portion of curve between bounds.
hold on
plot1(2)=plot(En,upp,'k');
plot1(3)=plot(En,low,'k');
plot1(4)=plot(En,bands);
plot1(6) = plot(En,gaps);
plot1(5)=plot(En,Eigval,'b');

% Set line widths and use custom colors.
set(plot1(1),'LineWidth',2,'Color',[1 0 0]);
set(plot1(2),'LineWidth',2,'Color',[0 0 0]);
set(plot1(3),'LineWidth',2,'Color',[0 0 0]);
set(plot1(5),'LineWidth',2,'Color',[0 0 1]);
set(plot1(4),'LineWidth',10,'Color',[.5 .5 1]);
set(plot1(6),'LineWidth',10,'Color',[1 .66 .66]);

% Define labels and axes size.
xlim([begin finish])
set(gca,'FontSize',12,'fontWeight','bold')
title('Conditions for Zero Eigenstate (in Dark/Blue Bands)')
xlabel('Energy (Joules)')
```

kp_wave_n.m:

```
% kp_wave_n.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Eigenfunctions for Negative Eigenvalues

clc; clear;

V0=1000;         % Potential Depth (eV)
a = 1e-10;       % Lengths a and b (m)
b = 1e-10;
m = 9.109e-31;  % Mass of electron (kg)
N = 10;           % Number of periods in crystal
n = 1;           % Propagation constant, determines phase of wavefunction
h = 6.626e-34 / (2*pi); % Planck's Constant:h_bar (J*s)
J_eV = 1.602e-19;       % J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

% Symbolic Variables, all simplified in terms of E:
syms E k q;
k = ((-2*m*E)^0.5)/h;
q = (2*m*(E+V0J))^0.5/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);

% Equation 33, the condition for negative Eigenvalues.
rhs = ( (k^2-q^2)/(2*k*q))*sinh(k*b)*sin(q*a)+cosh(k*b)*cos(q*a);
lhs=cos(Kab);

% Find Eigenvalues for a propagation constant, n.
[En,~] = true_intersect(rhs,lhs,E,-V0*J_eV,0,10000,1000,0,1)

% Select an Eigenvalue.
eig=5

% Generate M from equation 26.
iKab = 1i*K*(a+b);
M = [1                1                -1              -1               ;
     k               -k                -1i*q             1i*q           ;
     exp(-k*b+iKab)   exp(k*b+iKab)    -exp(1i*q*a)     -exp(-1i*q*a)    ;
     k*exp(-k*b+iKab) -k*exp(k*b+iKab) -1i*q*exp(1i*q*a) 1i*q*exp(-1i*q*a)]

%Substitute in the Eigenvalue, eig, for symbolic E.
Mnum=subs(M, E, En(eig));

% Convert M to Reduced Row Echelon Form. The tolerance for detecting
% negligible column elements may need to be reduced, if the trivial
% identity matrix solution is returned. See MATLAB documentation.
tol=(max(size(Mnum))*eps*norm(Mnum,inf))
R=rref(Mnum,tol)
Mrank=rank(Mnum)
```

```
% Substitute the Eigenvalue into the variables k and q.
kn=subs(k,E,En(eig));
qn=subs(q,E,En(eig));

% If rank 3, no further equations needed. Solve A,B,C in terms of D.
if Mrank==3
    syms A B C D
    A=-(B*R(1,2)+C*R(1,3)+D*R(1,4));
    B=-(C*R(2,3)+D*R(2,4));
    A=eval(char(A));
    C=-D*R(3,4)
    A=eval(char(A))
    B=eval(char(B))

% If rank 2, solve for A and B in terms of C and D. Solve equation 19 for C
% in terms of D
elseif Mrank == 2
    syms A B C D;
    A=-R(1,3)*C-R(1,4)*D
    B=-R(2,3)*C-R(2,4)*D
    C=solve(A+B-C-D,C)
    A=eval(char(A));
    B=eval(char(B));

% If rank 1, solve for B in terms of C and D. Solve equation 19 for A in
% terms of C and D, then equation 21 for C in terms of D.
elseif Mrank==1
    syms A B C D
    B=-R(1,3)*C-R(1,4)*D
    A=solve(A+B-C-D,A)
    C=solve(A*(kn)-(kn)*B-C*i*qn+D*i*qn,C)
    A=eval(char(A))
    B=eval(char(B))

end

% Divide A, B, and C by D and convert to double.
AofD=double(A/D);
BofD=double(B/D);
CofD=double(C/D);

% Normalize and solve for D.
xb=linspace(-b,0,100000*b/(a+b));
xa=linspace(0,a,100000*a/(a+b));
func1=AofD*exp(kn*xb)+BofD*exp(-kn*xb);
func2=CofD*exp(1i*qn*xa)+exp(-1i*qn*xa);
Dnew=(trapz([xb xa],[abs(func1).^2 abs(func2).^2])*N)^-1;

% Substitute D into A, B, and C.
Anew=subs(A,D,Dnew);
Bnew=subs(B,D,Dnew);
Cnew=subs(C,D,Dnew);
clear A B C D;
A=Anew(1);
```

```
B=Bnew(1);
C=Cnew(1);
D=Dnew;

% Position indices in regions 1 and 2:
x1=linspace(-b,0,(b/a+b)*10000);
x2=linspace(0,a,(a/a+b)*10000);
x3=linspace(a,a+b,(b/a+b)*10000);
x4=linspace(a+b,2*a+b,(a/a+b)*10000);

% Wavefunctions in regions 1 and 2:
psi1=A*exp(kn*x1)+B*exp(-kn*x1);
psi2=C*exp(1i*qn*x2)+D*exp(-1i*qn*x2);
psi3=(A*exp(kn*x1)+B*exp(-kn*x1))*exp(iKab);
psi4=(C*exp(1i*qn*x2)+D*exp(-1i*qn*x2))*exp(iKab);

% Plot real or imaginary part or probability density function.
plot_type='imag' %'real, 'imag', or 'prob'

%Plot wavefunction in regions 1 & 2
if plot_type == 'prob'
    plot1 = plot([x1 x2 x3 x4],[abs(psi1).^2 abs(psi2).^2 abs(psi3).^2
abs(psi4).^2], 'g');
    set(plot1,'Color',[0 0.5 0]);
elseif plot_type == 'imag'
    plot1 = plot([x1 x2 x3 x4], [imag(psi1) imag(psi2) imag(psi3)
imag(psi4)],'r');
elseif plot_type == 'real'
    plot1 = plot([x1 x2 x3 x4], [real(psi1) real(psi2) real(psi3)
real(psi4)]);
end
set(plot1,'LineWidth',2);

% Set line width and labels.
grid on;
set(gca,'FontSize',12,'fontWeight','bold')
xlabel({'distance (m)'});
if plot_type == 'prob'
    title({'|\psi^-|^2 : Position Probability Density Function (E<0)'});
elseif plot_type == 'imag'
    title({'Imag(\psi^-) : Imaginary Part of Eigenfunction (E<0)'});
elseif plot_type == 'real'
    title({'Real(\psi^-) : Real Part of Eigenfunction (E<0)'});
end

% Verify normalization.
trapz([abs(psi1).^2 abs(psi2).^2 abs(psi3).^2 abs(psi4).^2])
```

kp_wave_p.m:

```
% kp_wave_p.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Eigenfunctions for Positive Eigenvalues

clc; clear;

V0=1000;         % Potential Depth (eV)
a = 1e-10;       % Lengths a and b (m)
b = 1e-10;
m = 9.109e-31;   % Mass of electron (kg)
N = 10;           % Number of periods in crystal
n = 3;           % Propagation constant, determines phase of wavefunction
h = 6.626e-34 / (2*pi); % Planck's Constant:h_bar (J*s)
J_eV = 1.602e-19;       % J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

%Symbolic Variables, all simplified in terms of E:
syms E k q;
u = ((2*m*E)^0.5)/h;
w = (2*m*(E+V0J))^0.5/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);

% Equation 58, the condition for positive Eigenvalues.
rhs = ( -(u^2+w^2)/(2*u*w))*sin(u*b)*sin(w*a)+cos(u*b)*cos(w*a);
lhs=cos(Kab);

% Find Eigenvalues for a propagation constant, n.
[En,~] = true_intersect(rhs,lhs,E,-1e-17,V0J,100000,1000,0,1)

% Select an Eigenvalue.
eig=2

my_En = En(2)/J_eV

% Generate Q from equation 53.
iKab = 1i*K*(a+b);
Q = [     1                      0                    -1              0      ;
          0                      u                     0             -w      ;
  -cos(u*b)*exp(iKab)    sin(u*b)*exp(iKab)        cos(w*a)      sin(w*a)   ;
 -u*sin(u*b)*exp(iKab)   -u*cos(u*b)*exp(iKab)   -w*sin(w*a)    w*cos(w*a)]

% Substitute in the Eigenvalue, eig, for symbolic E.
Qnum=subs(Q, E, En(eig));


% Convert Q to Reduced Row Echelon Form.
tol=(max(size(Qnum))*eps *norm(Qnum,inf))*50
R=rref(Qnum,tol)
```

```
Qrank=rank(R)

% Substitute the Eigenvalue into the variables u and w.
un=subs(u,E,En(eig));
wn=subs(w,E,En(eig));

% If rank 3, no further equations needed. Solve F, G, J in terms of L.
if Qrank==3
    syms F G J L
    F=-(G*R(1,2)+J*R(1,3)+L*R(1,4));
    G=-(J*R(2,3)+L*R(2,4));
    F=eval(char(F));
    J=-L*R(3,4)
    F=eval(char(F))
    G=eval(char(G))

% If rank 2, solve for F and G in terms of J and L. Solve equation 50 for J
% in terms of L.
elseif Qrank == 2
    syms F G J L;
    F=-R(1,3)*J-R(1,4)*L
    G=-R(2,3)*J-R(2,4)*L
    J=solve(-F*cos(un*b)*exp(iKab)+G*sin(un*b)*exp(iKab)...
        +J*cos(wn*a)+L*sin(wn*a),J)
    F=eval(char(F))
    G=eval(char(G))

% The rank 1 case was not occurring in recent testing, so this method
% couldn't be verified.
%elseif Qrank==1
%    syms F G J L
%    F=-R(1,2)*G-R(1,3)*J-R(1,4)*L
%    J=solve(F-J,J)
%    F=eval(char(F))
%    G=solve(-F*cos(un*b)*exp(iKab)+G*sin(un*b)*exp(iKab)...
%        +J*cos(wn*a)+L*sin(wn*a),G)
%    J=eval(char(F))
%    F=eval(char(F))
%    G=eval(char(F))
end

% Divide F, G, and J by L and convert to double.
FofL=double(F/L);
GofL=double(G/L);
JofL=double(J/L);

% Normalize and solve for L.
xb=linspace(-b,0,100000*b/(a+b));
xa=linspace(0,a,100000*a/(a+b));
func1=FofL*cos(un*xb)+GofL*sin(un*xb);
func2=JofL*cos(wn*xa)+sin(wn*xa);
Lnew=(trapz([xb xa],[abs(func1).^2 abs(func2).^2])*N)^-1;

% Substitute L into F, G, and J.
Fnew=subs(F,L,Lnew);
```

```
Gnew=subs(G,L,Lnew);
Jnew=subs(J,L,Lnew);
clear F G J L;
F=Fnew(1);
G=Gnew(1);
J=Jnew(1);
L=Lnew;

%Position indices in regions 1 and 2:
x1=linspace(-b,0,(b/a+b)*10000);
x2=linspace(0,a,(a/a+b)*10000);
x3=linspace(a,a+b,(b/a+b)*10000);
x4=linspace(a+b,2*a+b,(a/a+b)*10000);

%Wavefunctions in regions 1 and 2:
psi1=F*cos(un*x1)+G*sin(un*x1);
psi2=J*cos(wn*x2)+L*sin(wn*x2);
psi3=(F*cos(un*x1)+G*sin(un*x1))*exp(iKab);
psi4=(J*cos(wn*x2)+L*sin(wn*x2))*exp(iKab);

% Plot real or imaginary part or probability density function.
plot_type='imag' %'real, 'imag', or 'prob'

% Plot wavefunction in regions 1 & 2
if plot_type == 'prob'
    plot1 = plot([x1 x2 x3 x4],[abs(psi1).^2 abs(psi2).^2 abs(psi3).^2
abs(psi4).^2], 'g');
    set(plot1,'Color',[0 0.5 0]);
elseif plot_type == 'imag'
    plot1 = plot([x1 x2 x3 x4], [imag(psi1) imag(psi2) imag(psi3)
imag(psi4)],'r');
elseif plot_type == 'real'
    plot1 = plot([x1 x2 x3 x4], [real(psi1) real(psi2) real(psi3)
real(psi4)]);
end
set(plot1,'LineWidth',2);

% Set line width and labels.
grid on;
set(gca,'FontSize',12,'fontWeight','bold')
xlabel({'distance (m)'});
if plot_type == 'prob'
    title({'|\psi^+|^2 : Position Probability Density Function (E>0)'});
elseif plot_type == 'imag'
    title({'Imag(\psi^+) : Imaginary Part of Eigenfunction (E>0)'});
elseif plot_type == 'real'
    title({'Real(\psi^+) : Real Part of Eigenfunction (E>0)'});
end

% Verify normalization.
trapz([abs(psi1).^2 abs(psi2).^2 abs(psi3).^2 abs(psi4).^2])
```

kp_3wave.m

```
% kp_wave_n.m
% Daniel Landers
% PHY 381 Quantum Mechanics: Kronig-Penney Model
% Plotting Eigenfunctions for Negative Eigenvalues

clc; clear;

V0=1.692049204920492e+02;        % Potential Depth (eV)
a = 1e-10;      % Lengths a and b (m)
b = 1e-10;
m = 9.109e-31;  % Mass of electron (kg)
N = 10;          % Number of periods in crystal
n = 3;          % Propagation constant, determines phase of wavefunction
h = 6.626e-34 / (2*pi); % Planck's Constant:h_bar (J*s)
J_eV = 1.602e-19;       % J/eV for eV-->J conversion

% Convert potential depth to joules.
V0J=V0*J_eV;

% Symbolic Variables, all simplified in terms of E:
syms E k q l u w;
k = ((-2*m*E)^0.5)/h;
q = (2*m*(E+V0J))^0.5/h;
l = ((2*m*E)^0.5)/h;
u = ((2*m*E)^0.5)/h;
w = (2*m*(E+V0J))^0.5/h;
K = (2*pi*n)/(N*(a+b));
Kab = K*(a+b);

% Equation 33, the condition for negative Eigenvalues.
rhs_n = ( (k^2-q^2)/(2*k*q))*sinh(k*b)*sin(q*a)+cosh(k*b)*cos(q*a);
lhs = cos(Kab);

% Equation 83, the criterion for an Eigenvalue of zero.
rhs_eq = ( cos(l*a)-(l*b/2)*sin(l*a));

% Equation 58, the condition for positive Eigenvalues.
rhs_p = ( -(u^2+w^2)/(2*u*w))*sin(u*b)*sin(w*a)+cos(u*b)*cos(w*a);

% Find Eigenvalues for a propagation constant, n.
[En_n,~] = true_intersect(rhs_n,lhs,E,-V0*J_eV,0,10000,1000,0,1)

%Find Eigenvalues that meet criterion, for propogation constant, n.
[En_eq,~] = true_intersect(rhs_eq,lhs,E,0,10000*J_eV,10000,1000,0,1)

% Find Eigenvalues for a propagation constant, n.
[En_p,~] = true_intersect(rhs_p,lhs,E,-1e-17,V0J,100000,1000,0,1)

% Select an Eigenvalue.
eig_n=3
eig_eq = 3
eig_p = 1
```

```
% Generate M from equation 26 and R from equation 78.
iKab = 1i*K*(a+b);
M = [1                 1                 -1               -1               ;
     k                -k                 -1i*q              1i*q            ;
     exp(-k*b+iKab)    exp(k*b+iKab)     -exp(1i*q*a)      -exp(-1i*q*a)    ;
     k*exp(-k*b+iKab) -k*exp(k*b+iKab) -1i*q*exp(1i*q*a) 1i*q*exp(-1i*q*a)]

R = [0                 1                 -1               0              ;
     1                 0                  0              -l           ;
     -b*exp(iKab)    exp(iKab)         -cos(l*a)       -sin(l*a)    ;
       exp(iKab)       0              l*sin(l*a)     -l*cos(l*a) ]

Q = [     1                       0                     -1              0      ;
          0                       u                      0             -w      ;
  -cos(u*b)*exp(iKab)     sin(u*b)*exp(iKab)         cos(w*a)      sin(w*a)  ;
 -u*sin(u*b)*exp(iKab)    -u*cos(u*b)*exp(iKab)    -w*sin(w*a)    w*cos(w*a)]

%Substitute in the Eigenvalue, eig, for symbolic E.
Mnum=subs(M, E, En_n(eig_n));
Rnum=subs(R, E, En_eq(eig_eq))
Qnum=subs(Q, E, En_p(eig_p));


% Convert M to Reduced Row Echelon Form. The tolerance for detecting
% negligible column elements may need to be reduced, if the trivial
% identity matrix solution is returned. See MATLAB documentation.
tol=(max(size(Mnum))*eps*norm(Mnum,inf))
Mrref=rref(Mnum,tol)
Mrank=rank(Mnum)
Rrref=rref(Rnum,tol)
Rrank=rank(Rrref)
Qrref=rref(Qnum,tol)
Qrank=rank(Qrref)

% Substitute the Eigenvalue into the variables k, q, l, u and w.
kn=subs(k,E,En_n(eig_n));
qn=subs(q,E,En_n(eig_n));
ln=subs(l,E,En_eq(eig_eq));
un=subs(u,E,En_p(eig_p));
wn=subs(w,E,En_p(eig_p));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Negative Eigenfunctions: Finding constants A, B, C, and D

% If rank 3, no further equations needed. Solve A,B,C in terms of D.
if Mrank==3
    syms A B C D
    A=-(B*Mrref(1,2)+C*Mrref(1,3)+D*Mrref(1,4));
    B=-(C*Mrref(2,3)+D*Mrref(2,4));
    A=eval(char(A));
    C=-D*Mrref(3,4)
    A=eval(char(A))
    B=eval(char(B))
```

```
% If rank 2, solve for A and B in terms of C and D. Solve equation 19 for C
% in terms of D
elseif Mrank == 2
    syms A B C D;
    A=-Mrref(1,3)*C-Mrref(1,4)*D
    B=-Mrref(2,3)*C-Mrref(2,4)*D
    C=solve(A+B-C-D,C)
    A=eval(char(A));
    B=eval(char(B));

% If rank 1, solve for B in terms of C and D. Solve equation 19 for A in
% terms of C and D, then equation 21 for C in terms of D.
elseif Mrank==1
    syms A B C D
    B=-Mrref(1,3)*C-Mrref(1,4)*D
    A=solve(A+B-C-D,A)
    C=solve(A*(kn)-(kn)*B-C*i*qn+D*i*qn,C)
    A=eval(char(A))
    B=eval(char(B))

end

% Divide A, B, and C by D and convert to double.
AofD=double(A/D);
BofD=double(B/D);
CofD=double(C/D);

% Normalize and solve for D.
xb=linspace(-b,0,100000*b/(a+b));
xa=linspace(0,a,100000*a/(a+b));
func1_n=AofD*exp(kn*xb)+BofD*exp(-kn*xb);
func2_n=CofD*exp(1i*qn*xa)+exp(-1i*qn*xa);
Dnew=(trapz([xb xa],[abs(func1_n).^2 abs(func2_n).^2])*N)^-1;

% Substitute D into A, B, and C.
Anew=subs(A,D,Dnew);
Bnew=subs(B,D,Dnew);
Cnew=subs(C,D,Dnew);
clear A B C D;
A=Anew(1);
B=Bnew(1);
C=Cnew(1);
D=Dnew;

% Position indices in regions 1 and 2:
x1_n=linspace(-b,0,(b/a+b)*10000);
x2_n=linspace(0,a,(a/a+b)*10000);
x3_n=linspace(a,a+b,(b/a+b)*10000);
x4_n=linspace(a+b,2*a+b,(a/a+b)*10000);

% Wavefunctions in regions 1 and 2:
psi1_n=A*exp(kn*x1_n)+B*exp(-kn*x1_n);
psi2_n=C*exp(1i*qn*x2_n)+D*exp(-1i*qn*x2_n);
psi3_n=(A*exp(kn*x1_n)+B*exp(-kn*x1_n))*exp(iKab);
psi4_n=(C*exp(1i*qn*x2_n)+D*exp(-1i*qn*x2_n))*exp(iKab);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eigenvalue of Zero: Solving for constants, alpha, beta, gamma, epsilon

if Rrank == 2
    syms alp bet gam eps;

    %Define alp and bet
    alp=-Rrref(1,3)*gam-Rrref(1,4)*eps
    bet=-Rrref(2,3)*gam-Rrref(2,4)*eps
    gam=solve(bet-gam,gam)
    alp=eval(char(alp))
    bet=eval(char(bet))

elseif Rrank==3
    syms alp bet gam eps
    alp=-(bet*Rrref(1,2)+gam*Rrref(1,3)+eps*Rrref(1,4));
    bet=-(gam*Rrref(2,3)+eps*Rrref(2,4));
    alp=eval(char(alp));
    gam=-eps*Rrref(3,4)
    alp=eval(char(alp))
    bet=eval(char(bet))
end

alp_eps=double(alp/eps);
bet_eps=double(bet/eps);
gam_eps=double(gam/eps);

func1_eq=alp_eps*xb+bet_eps;
func2_eq=gam_eps*cos(ln*xa)+sin(ln*xa);
eps_new=(trapz([xb xa],[abs(func1_eq).^2 abs(func2_eq).^2])*N)^-1;

alp_new=subs(alp,eps,eps_new);
bet_new=subs(bet,eps,eps_new);
gam_new=subs(gam,eps,eps_new);
clear alp bet gam eps;

alp=alp_new(1);
bet=bet_new(1);
gam=gam_new(1);
eps=eps_new;

x1_eq=linspace(-b,0,(b/a+b)*10000);
x2_eq=linspace(0,a,(a/a+b)*10000);
x3_eq=linspace(a,a+b,(b/a+b)*10000);
x4_eq=linspace(a+b,2*a+b,(a/a+b)*10000);

psi1_eq=alp*x1_eq+bet;
psi2_eq=gam*cos(ln*x2_eq)+eps*sin(ln*x2_eq);
psi3_eq=(alp*x1_eq+bet)*exp(iKab);
psi4_eq=(gam*cos(ln*x2_eq)+eps*sin(ln*x2_eq))*exp(iKab);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Positive Eigenfunctions: Solving for constants F, G, J, and L.

% If rank 3, no further equations needed. Solve F, G, J in terms of L.
if Qrank==3
    syms F G J L
    F=-(G*Qrref(1,2)+J*Qrref(1,3)+L*Qrref(1,4));
    G=-(J*Qrref(2,3)+L*Qrref(2,4));
    F=eval(char(F));
    J=-L*Qrref(3,4)
    F=eval(char(F))
    G=eval(char(G))

% If rank 2, solve for F and G in terms of J and L. Solve equation 50 for J
% in terms of L.
elseif Qrank == 2
    syms F G J L;
    F=-Qrref(1,3)*J-Qrref(1,4)*L
    G=-Qrref(2,3)*J-Qrref(2,4)*L
    J=solve(-F*cos(un*b)*exp(iKab)+G*sin(un*b)*exp(iKab)...
        +J*cos(wn*a)+L*sin(wn*a),J)
    F=eval(char(F))
    G=eval(char(G))
end

% Divide F, G, and J by L and convert to double.
FofL=double(F/L);
GofL=double(G/L);
JofL=double(J/L);

% Normalize and solve for L.
func1_p=FofL*cos(un*xb)+GofL*sin(un*xb);
func2_p=JofL*cos(wn*xa)+sin(wn*xa);
Lnew=(trapz([xb xa],[abs(func1_p).^2 abs(func2_p).^2])*N)^-1;

% Substitute L into F, G, and J.
Fnew=subs(F,L,Lnew);
Gnew=subs(G,L,Lnew);
Jnew=subs(J,L,Lnew);
clear F G J L;
F=Fnew(1);
G=Gnew(1);
J=Jnew(1);
L=Lnew;

%Position indices in regions 1 and 2:
x1_p=linspace(-b,0,(b/a+b)*10000);
x2_p=linspace(0,a,(a/a+b)*10000);
x3_p=linspace(a,a+b,(b/a+b)*10000);
x4_p=linspace(a+b,2*a+b,(a/a+b)*10000);

%Wavefunctions in regions 1 and 2:
psi1_p=F*cos(un*x1_p)+G*sin(un*x1_p);
psi2_p=J*cos(wn*x2_p)+L*sin(wn*x2_p);
psi3_p=(F*cos(un*x1_p)+G*sin(un*x1_p))*exp(iKab);
psi4_p=(J*cos(wn*x2_p)+L*sin(wn*x2_p))*exp(iKab);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plots


% Plot real or imaginary part or probability density function.
plot_type='imag' %'real, 'imag', or 'prob'



%Plot wavefunction in regions 1 & 2
if plot_type == 'prob'
    subplot(3,1,1)
    plot1 = plot([x1_n x2_n x3_n x4_n],[abs(psi1_n).^2 abs(psi2_n).^2
abs(psi3_n).^2 abs(psi4_n).^2], 'g');
    title({'|\psi^-|^2 : Position Probability Density Function (E<0)'});
    subplot(3,1,2)
    plot2 = plot([x1_eq x2_eq x3_eq x4_eq],[abs(psi1_eq).^2 abs(psi2_eq).^2
abs(psi3_eq).^2 abs(psi4_eq).^2], 'g');
    subplot(3,1,3)
    plot3 = plot([x1_p x2_p x3_p x4_p],[abs(psi1_p).^2 abs(psi2_p).^2
abs(psi3_p).^2 abs(psi4_p).^2], 'g');
    set(plot1,'Color',[0 0.5 0]);
    set(plot2,'Color',[0 0.5 0]);
    set(plot3,'Color',[0 0.5 0]);
elseif plot_type == 'imag'
    subplot(3,1,1)
    plot1 = plot([x1_n x2_n x3_n x4_n], [imag(psi1_n) imag(psi2_n)
imag(psi3_n) imag(psi4_n)],'r');
    title({'Imag(\psi^-) : Imaginary Part of Eigenfunction (E<0)'});
    subplot(3,1,2)
    plot2 = plot([x1_eq x2_eq x3_eq x4_eq], [imag(psi1_eq) imag(psi2_eq)
imag(psi3_eq) imag(psi4_eq)],'r');
    subplot(3,1,3)
    plot3 = plot([x1_p x2_p x3_p x4_p], [imag(psi1_p) imag(psi2_p)
imag(psi3_p) imag(psi4_p)],'r');
    set(plot1,'Color',[0.5 0 0]);
    set(plot2,'Color',[0.5 0 0]);
    set(plot3,'Color',[0.5 0 0]);
elseif plot_type == 'real'
    subplot(3,1,1)
    plot1 = plot([x1_n x2_n x3_n x4_n], [real(psi1_n) real(psi2_n)
real(psi3_n) real(psi4_n)]);
    title({'Real(\psi^-) : Real Part of Eigenfunction (E<0)'});
    subplot(3,1,2)
    plot2 = plot([x1_eq x2_eq x3_eq x4_eq], [real(psi1_eq) real(psi2_eq)
real(psi3_eq) real(psi4_eq)]);
    subplot(3,1,3)
    plot3 = plot([x1_p x2_p x3_p x4_p], [real(psi1_p) real(psi2_p)
real(psi3_p) real(psi4_p)]);

    set(plot1,'Color',[0 0 0.5]);
    set(plot2,'Color',[0 0 0.5]);
    set(plot3,'Color',[0 0 0.5]);
```

```
end
set(plot1,'LineWidth',2);
set(plot2,'LineWidth',2);
set(plot3,'LineWidth',2);

% Set line width and labels.
grid on;
set(gca,'FontSize',10,'fontWeight','bold')
xlabel({'distance (m)'});


% Verify normalization.
%trapz([abs(psi1).^2 abs(psi2).^2 abs(psi3).^2 abs(psi4).^2])
```

true_intersect.m

```
%true_intersect
%Finds the intersection between two symbolic functions with respect to an
%independent variable, over a defined range, and precision. Also, has an
%option to plot the functions and their intersections.
%Written by Daniel Landers 2/7/14
%Updated 6/11/14
%
% Syntax:
%   [index,isectVal] =
true_intersect(func1,func2,var,start,finish,prec1,prec2,plot_on)
%
% Inputs:
%  func1 - first function
%  func2 - second function
%  var - independent variable of functions
%  start - left boundary of search range
%  finish - right boundary of search range
%  prec1 - Coarse precision/sensitivity, # of samples in range
%  prec2 - Fine precision, # of samples between neighboring coarse samples
%  plot_on - Enables plotting when  1
%
% Outputs:
%  index - Independent variable at intersection. Not really an index!
%  isectVal - Function value at intersection
function [index,isectVal] =
true_intersect(func1,func2,var,start,finish,prec1,prec2,plot_on,realImag)


%Generate vector of Independent Variable with elements defined
%by start, end, and precision
x=linspace(start,finish,prec1);
%Substitute symbolic expressions with numeric result of expression of x
eq1 = subs(func1,var,{x});
eq2 = subs(func2,var,{x});
```

```
%But if either of the functions are just a constant, multiply it by an
%array the size of x.
if length(eq1)==1
    eq1=eq1*ones(1,prec1);
end
if length(eq2)==1
    eq2=eq2*ones(1,prec1);
end

if realImag==1        %Added support to force real or complex
    eq1=real(eq1);
    eq2=real(eq2);
elseif realImag==2
    eq1=complex(eq1);
    eq2=complex(eq2);
end
%Detects when "intersections" occur, including asymptotes
detect = diff(sign(eq1-eq2));

%Finds indices of intersections
isect = find(detect);
clear detect;
j = 1;
for i = 1:length(isect)
    %If intersection occurs on first sample then we can only zoom
    %in forward, otherwise we zoom in between previous and next.
    if isect(i)==1
        xZoom = linspace(x(isect(i)),x(isect(i)+1),prec2);
        scl = (x(isect(i)+1)-x(isect(i)))/prec2;
    else
        xZoom = linspace(x(isect(i)-1),x(isect(i)+1),prec2);
        scl = (x(isect(i)+1)-x(isect(i)-1))/prec2;
    end
    %Again, convert to vector, and account for constant expressions
    eq1z = subs(func1,var,{xZoom});
    eq2z = subs(func2,var,{xZoom});
    if length(eq1z)==1
        eq1z=eq1z*ones(1,prec2);
    end
    if length(eq2z)==1
        eq2z=eq2z*ones(1,prec2);
    end
    if realImag==1         %Added Support to force real or complex!
        eq1z=real(eq1z);
        eq2z=real(eq2z);
    elseif realImag==2
        eq1z=complex(eq1z);
        eq2z=complex(eq2z);
    end
    %Find the sign of the difference in slopes...
    %(By slopes I mean the difference between adjacent elements--diff() )
    slop(i,:) = sign(diff(eq1z) - diff(eq2z));
    %Shift them ahead. We seek the sign of the difference in slopes just
    %before intersection.
    slop(i,:) = circshift(slop(i,:),[0 1]);
    %Look for intersections, then their indices
```

```
    detectz(i,:) = diff(sign(eq1z-eq2z));
    isectz = find(detectz(i,:));
    %If there are more than one intersections when zoomed in, it
    %might be good to zoom farther to catch all intersections.
    %COMMENT OUT ERROR MESSAGE IF YOU WANT TO USE THE FIRST INTERSECTION
%    if length(isectz) > 1
%        error('The coarse resolution  is too small.', ...
%        'Multiple intersections were detected.')
%    end
%
%If the difference between adjacent elements of the (sign of the difference
%of the two functions) is NOT EQUAL TO the sign of the difference of
%slopes, then the intersection is actually an asymptote, --NaN.
    if sign(detectz(i,:))~=slop(i,:)
        isect2(i)=NaN;           %Don't record this index of asymptote
    elseif isect(i)==1           %Intersection on first sample
        isect2(i) = x(isect(i))+scl*(isectz(1));
        %Intersection Value
        trueIsect(j,1)=eq1z(isectz(1));
        %Intersection index, high precision
        x_n(j,1)=isect2(i);
        j=j+1;
    else
        isect2(i) = x(isect(i)-1)+scl*(isectz(1));
        trueIsect(j,1)=eq1z(isectz(1));
        x_n(j,1)=isect2(i);
        j=j+1;
    end
end
index = x_n;                 %outputs
isectVal = trueIsect;
if plot_on==1                %If using plot option...
    hold off;
    plot(x,eq1,'b');
    hold on;
    plot(x,eq2, 'r');
    plot(x_n,trueIsect,'MarkerFaceColor',[1 1
0],'MarkerSize',10,'Marker','pentagram',...
    'LineStyle','none',...
    'Color',[0 0 0]);
end
end
```