Bound States and Scattering States of a Quantum Particle on Half-Infinite Lattices

by

AARON HSU

Faculty Mentor: Bruno Nachtergaele, Ph.D.

SENIOR THESIS

Submitted in partial satisfaction of the requirements for Highest Honors for the degree of

BACHELOR OF ARTS AND SCIENCES

in

MATHEMATICS AND STATISTICS

in the

COLLEGE OF LETTERS AND SCIENCE

of the

UNIVERSITY OF CALIFORNIA, DAVIS

June 12, 2014

Contents

Abstract

The motion of a quantum particle, such as an electron, obeys the Schrödinger equation, as determined by its Hamiltonian (which corresponds to the total energy of the particle). Such a particle may either remain localized in a small region of space (a bound state) or diffuse through the entire system (a scattering state). We are particularly interested in two-dimensional lattice systems with a boundary, such as a lattice of points in half of the xy-plane; physically, this might correspond to a thin layer of atoms on a substrate or to the surface of a crystal. Recent research has revealed that a particle in such a system may either be confined to the boundary in a bound state or diffuse along the boundary. We extend this knowledge by analyzing two configurations that until now have not been studied. Applying techniques of spectral analysis, linear algebra, and functional analysis, we investigate conditions differentiating the two cases and test a conjecture for a general criterion for the existence of a bound state using spectral gaps. We anticipate that our results may find practical applications in the fields of condensed matter physics and materials science and in the design of quantum information devices.

Keywords: *infinite lattices, bound state, scattering state, discrete Laplacian, spectral gap, transfer matrix, boundary conditions, perturbation*

1 Introduction

Throughout human history, man has attempted to explain the universe around him, and the progress of much of science, in particular much of physics, can be characterized by attempts to describe the world at an increasingly smaller scale. As a result, to this day, a significant portion of science is dedicated to characterizing matter and describing the manners in which it behaves.

Over the course of time, science has developed successively more refined theories of matter. In 1803, for instance, John Dalton formulated a theory that all matter is composed of atoms. This was followed by discovery of subatomic particles by Pierre and Marie Curie and electrons by J.J. Thomson in 1898. Subsequently, around the turn of the 19th century, Max Planck theorized as to what are now called photons, or quantized packets of light, the particle nature of which was explored by Albert Einstein a few years later.

Following this, Ernest Rutherford demonstrated in 1909 the existence of nuclei in atoms, a model later refined by Neils Bohr in 1913. In 1931, James Chadwick discovered the neutron, and in 1964, Murray Gell Mann and George Zweig proposed the quark model, describing elementary particles even smaller than the subatomic particles of proton and neutron. Recent research has attempted to describe yet smaller components of matter, leading to the development of, e.g., string theory and the theory of loop quantum gravity.

In this paper, however, the matters we are concerned with shall not be quite so small. We consider the kinetic energy of a quantum particle, such as a quark or a lepton, that moves on a system modeled as a finite or infinite integer lattice. In particular, we are interested in two-dimensional lattice systems with a boundary, such as a lattice of points in half of the xy-plane. Physically, such a system might correspond to a thin layer of atoms on a substrate or to the surface of a crystal.

Quantum particles are among the basic building blocks of matter. These particles include the well-known electron, which is an example of a lepton, as well as less well-known leptons, such as muons and neutrinos, in addition to photons, various quarks, and the recently-discovered Higgs boson. The dynamics of such a quantum particle at low energies, i.e., in the non-relativistic regime, are described by the celebrated Schrödinger's equation:

$$i\hbar\frac{d}{dt}\psi = H\psi.$$

This equation can be solved by diagonalizing the operator H [2]:

$$\psi_t = e^{-itH}\psi_0.$$

This operator is called the Hamiltonian operator and represents interactions between particles in the system; it can be thought of as a sum of matrices, each representing an interaction between two particles or the kinetic energy of a single particle. Quantum particles can reside in one of possibly many energy levels, which are given by the eigenstates of the Hamiltonian.

Quantum particles belonging to a lattice have associated wavevectors residing in a Hilbert space. Specifically, for any given lattice Λ with possibly multiple particles belonging to it, the Hilbert space of its wavevectors is

$$\mathcal{H}_\Lambda = \bigotimes_{x\in\Lambda} \mathbb{C}^2.$$

In particular, for a one-particle subspace, which we will restrict our attention to in this paper, the Hilbert space is reduced to

$$\mathcal{H}_\Lambda^{(1)} = \ell^2(\Lambda) \cong \mathcal{C}^{|\Lambda|},$$

where the ℓ^2 space is given by

$$\ell^2(\Lambda) = \{(z_\alpha)_{\alpha\in\Lambda} | z_\alpha \in \mathbb{C}, \sum_{\alpha\in\Lambda} |z_\alpha|^2 < \infty\}$$

with the inner product

$$\langle (z_\alpha), (z'_\alpha)\rangle = \sum_{\alpha\in\Lambda} \overline{z}_\alpha z'_\alpha.$$

The Hamiltonian we consider in this paper will initially be the discrete second-order Laplacian operator, which describes the kinetic energy of quantum particles:

Definition 1.1. *The second-order Laplacian operator is defined as follows:*

$$(H_\Lambda \psi)_x = \sum_{y,|y-x|=1} (\psi_x - \psi_y)$$

where Λ is a finite or infinite lattice, x and y denote vertices on Λ, and ψ denotes the wavevector describing the energy configuration of a particle on the lattice.

As can be seen from the expression for the operator, we shall only be concerned with nearest-neighbor interactions, i.e., interactions between lattice sites immediately adjoining one another, in addition to each lattice site's inherent energy. This shall remain the case throughout this paper, although we will generalize the relative weights of different interactions in Section 5.

As a remark on notation, in this paper, we will use ψ_x and $\psi(x)$ interchangeably to denote the x component of the wavevector ψ, depending on which notation is more clear.

This remainder of this paper is organized as follows. In Section 2, we will consider and solve completely the simplest non-trivial model, the finite one-dimensional ring. This model will be slightly modified in Section 3, where we will consider the one-dimensional free-end chain and solve the model by employing the technique of transfer matrices. We will also add a rank-one perturbation to the system and analyze how this changes the Hamiltonian operator's spectral

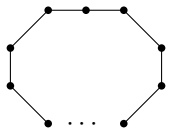
properties. In addition, we consider our first infinite models and analyze their bound states.

We shall then proceed to a two-dimensional lattice in Section 4, then extend it in Section 5 to the entire two-dimensional infinite lattice, which we analyze as a tensor structure of the one-dimensional case. In Section 6, we will state and prove a general theorem providing the existence of a unique zero ground state for general lattices of this type. Relying upon these results, in Section 7, we formulate a conjecture relating bound states of the Hamiltonians considered to their spectral gaps. We test this conjecture in Sections 8 and 9, using the results of Section 5 to inform our analytical and numerical analysis of two different half-infinite lattice models in two dimensions. We close with some discussion of our results and conclusions in the final section.

2 A Simple Example: The One-dimensional Ring

The first lattice Λ we consider is a finite, one-dimension joined-end chain of particles.

Figure 1: Finite One-dimension Ring



Our objective is to solve this model by deriving and diagonalizing the Hamiltonian corresponding to Λ, thereby finding its eigenstates and spectral characteristics. This simple example will also illustrate some basic techniques we will employ in solving subsequent models.

We begin by defining the set of basis vectors that will span the space of wavevectors (i.e., $\ell^2(\Lambda)$) for this matrix:

$$\{e_x\}_{x=1}^n \text{ where } (e_x)_y = \delta_{xy}. \tag{2.1}$$

From linear algebra, we know that for any N-dimensional inner product space V, the elements $(a_{jk})_{1\leq j,k\leq N}$ of any linear operator $A : V \to V$, with respect to an orthonormal basis $\{e_j\}$, are given by

$$a_{jk} = \langle e_j, Ae_k \rangle.$$

Applying this, we have the following proposition that gives our Hamiltonian:

Proposition 2.1. *The matrix corresponding to the discrete second-order Laplacian operator for the lattice Λ (i.e., a one-particle, one-dimensional joined-end*

chain) can be expressed as follows, in the given basis:

$$H_\Lambda = \begin{pmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & -1 \\ -1 & & & & -1 & 2 \end{pmatrix}. \tag{2.2}$$

Proof. By Definition 1, we have that

$$\begin{aligned} (H_\Lambda e_j)_x &= \sum_{y,|y-x|=1} ((e_j)_x - (e_j)_y) \\ &= \sum_{y,|y-x|=1} (\delta_{jx} - \delta_{jy}) \\ &= \begin{cases} 2 & \text{if } j = x \\ -1 & \text{if } j = x \pm 1 \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Multiplying a basis vector on the left then yields the elements of the matrix:

$$(H_\Lambda)_{jk}\, e_k H_\Lambda e_j = \begin{cases} 2 & \text{if } k = j \\ -1 & \text{if } k = j \pm 1 \\ 0 & \text{else.} \end{cases} \tag{2.3}$$

□

We can diagonalize this matrix with the Discrete Fourier Transform:

Theorem 2.1. *The eigenvalues λ_j and eigenvectors f_j of the Hamiltonian operator (2.2) are as follows:*

$$\lambda_j = 2 - 2\cos\left(\frac{2\pi i}{M} j\right) \quad \forall j = 1, \ldots, M \tag{2.4}$$

$$f_j = (1, \omega^{1j}, \omega^{2j}, \ldots, \omega^{(M-1)j}) \quad \forall j = 1, \ldots, M. \tag{2.5}$$

Proof. We observe that this matrix has identical rows that are merely shifted horizontally, i.e., it is a circulant matrix. This allows us to diagonalize it with the Discrete Fourier Transform (DFT) matrix [3]:

$$F_M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{1\cdot 1} & \omega^{1\cdot 2} & \cdots & \omega^{1\cdot(M-1)} \\ 1 & \omega^{2\cdot 1} & \omega^{2\cdot 2} & \cdots & \omega^{2\cdot(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(M-1)\cdot 1} & \omega^{(M-1)\cdot 2} & \cdots & \omega^{(M-1)\cdot(M-1)} \end{pmatrix},$$

where $\omega = e^{2\pi i/M}$ is the primitive M-th root of unity.

The diagonalization relation is as follows:

$$\begin{aligned} & F_M^{-1} H_\Lambda F_M = \Delta \\ \Leftrightarrow\ & H_\Lambda F_M = F_M \Delta \\ \Leftrightarrow\ & H_\Lambda f_j = \lambda_j f_j \ \ \forall j = 1, \ldots, M, \end{aligned}$$

writing $F_M = (f_1, f_2, \ldots, f_M)$ and $\Delta = diag(\lambda_m)$.

Therefore, the eigenvectors of H_Λ are given by the columns of the DFT matrix F_M, namely,

$$f_j = (1, \omega^{1j}, \omega^{2j}, \ldots, \omega^{(M-1)j}) \ \ \forall j = 1, \ldots, M.$$

The associated eigenvalues are then found as follows:

$$\begin{aligned} (\lambda_j f_j)_k &= (H_\Lambda f_j)_k \\ &= 2(f_j)_k - (f_j)_{k-1} - (f_j)_{k+1} \\ &= 2\omega^{jk} - \omega^{j(k-1)} - \omega^{j(k+1)}. \end{aligned}$$

We also have that

$$\begin{aligned} (\lambda_j f_j)_k &= \lambda_j (f_j)_k \\ &= \lambda_j \omega^{jk}. \end{aligned}$$

Setting these expressions equal, we obtain

$$\begin{aligned} \lambda_j \omega^{jk} &= 2\omega^{jk} - \omega^{j(k-1)} - \omega^{j(k+1)} \\ &= 2\omega^{jk} - \omega^{jk}\omega^{-j} - \omega^{jk}\omega^{j}. \end{aligned}$$

Hence,

$$\begin{aligned} \lambda_j &= 2 - \omega^{-j} - \omega^{j} \\ &= 2 - \cos\left(-\frac{2\pi i}{M} j\right) - i \sin\left(-\frac{2\pi i}{M} j\right) - \cos\left(\frac{2\pi i}{M} j\right) - i \sin\left(\frac{2\pi i}{M} j\right) \\ &= 2 - 2\cos\left(\frac{2\pi i}{M} j\right) \ \ \forall j = 1, \ldots, M \end{aligned}$$

are the respective associated eigenvalues for the eigenvectors f_j previously obtained. □

3 The One-dimensional Free-end Chain

The next class of models we consider are logical variants of the first, where the ends of the chain are no longer joined together (Figure 2). We first consider the basic case, i.e., simply removing the interactions between the edge vertices. We then introduce a perturbation to this model at the edge of the lattice.

Figure 2: A finite one-dimensional free-end lattice

3.1 The No-perturbation Model

Following the same method as in Proposition 2.1 above, it is easy to see that the Hamiltonian for this case is exactly like that in Eq. (2.2), but with no -1's on the lower-left and upper-right corners:

$$H = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}.$$

As a result of the removed elements, the matrix is no longer circulant, and thus, the Discrete Fourier Transform method used in Section 2 can longer be applied. Instead, we apply to the elements of the eigenvectors v_i an *Ansatz* of the form

$$(v_i)_n = \sin(\alpha n + \gamma),$$

verify that this produces eigenvectors, and solve for acceptable values of α and γ. The eigenvalues follow immediately.

Our results are given in the following theorem:

Theorem 3.1. *The Hamiltonian $\mathcal{H}_\Lambda$ for the one-particle, one-dimensional free-end lattice as described above has eigenvalues λ_i and eigenvectors $v_i = (v_i)_{n=1}^N$ as follows:*

$$\lambda_i = 2 - 2\cos(\gamma), \gamma = \frac{2\pi n}{N} \tag{3.1}$$

$$(v_i)_n = \sin((n-1)\alpha + \gamma). \tag{3.2}$$

Proof. For $n = 2, \ldots, N-1$, i.e., away from the ends of the lattice, we have

$$\begin{aligned}
(\lambda_i v_i)_n &= [2 - 2\cos(\gamma)]\sin(\alpha(n-1) + \gamma) \\
&= 2\sin(\alpha(n-1) + \gamma) \\
&\qquad - [\sin(\alpha(n-1) + b)\cos(\alpha) - \cos(\alpha(n-1) + b)\sin(\alpha)] \\
&\qquad - [\sin(\alpha(n-1) + \gamma)\cos(\alpha) + \cos(\alpha(n-1) + \gamma)\sin(\alpha)] \\
&= 2\sin(\alpha(n-1) + \gamma) - \sin(\alpha((n-1) - 1) + \gamma) \\
&\qquad - \sin(\alpha((n-1) + 1) + \gamma) \\
&= 2\sin(\alpha(n-1) + \gamma) - \sin(\alpha(n-2) + \gamma) - \sin(\alpha n + \gamma) \\
&= 2(v_i)_n - (v_i)_{n-1} - (v_i)_{n+1} \\
&= (\mathcal{H}_\Lambda v_i)_n.
\end{aligned}$$

where the third equality uses the sum-difference formula for the sine function.[1]

Note that so far, any values of α and γ are acceptable. These will be specified by the two boundary conditions.

For $n = 1$, the condition we must satisfy is $H_\Lambda(v_i)_1 = \lambda_i(v_i)_1$, or

$$2\sin(\gamma) - \sin(\alpha + \gamma) = [2 - 2\cos(\gamma)]\sin(\gamma) \tag{3.3}$$

This equation gives:

$$\begin{aligned} \sin(\alpha + \gamma) &= 2\cos(\gamma)\sin(\gamma) \\ \sin(\alpha + \gamma) &= \sin(2\gamma) \\ \sin\left(\alpha + \frac{2\pi n}{N}\right) &= \sin\left(2\frac{2\pi n}{N}\right) \end{aligned}$$

Therefore, on the boundaries, we have, for $n = 1$:

$$\begin{aligned} (\lambda_i v_i)_1 &= [2 - 2\cos(\gamma)]\sin(\gamma) \\ &= 2\sin(\gamma) - 2\cos(\gamma)\sin(\gamma) \\ &= 2\sin(\gamma) - \sin(2\gamma) \\ &= 2\sin(\gamma) - \sin(\alpha + \gamma) \\ &= 2(v_i)_1 - (v_i)_2 \\ &= (\mathcal{H}_\Lambda v_i)_1. \end{aligned}$$

The case for $n = N$ is analogous. □

3.2 The Edge-perturbation Model

Next, we add a rank-one perturbation to the model. The notation will be clearer if we generalize the Hamiltonian somewhat by replacing the 2's with a variable b. To add a perturbation, we further replace the upper-left-most b with a parameter a to determine how this affects the eigenstates of the Hamiltonian. (Physically, this might correspond, for example, to a one-atom impurity on a crystal.) The resulting matrix is as follows:

$$H_\Lambda = \begin{pmatrix} a & -1 & & & & \\ -1 & b & -1 & & & \\ & -1 & b & & & \\ & & & \ddots & & \\ & & & & b & -1 \\ & & & & -1 & b \end{pmatrix}.$$

[1] The sum-difference formula for sine is:

$$\sin(\alpha \pm \gamma) = \sin(\alpha)\cos(\gamma) \pm \cos(\alpha)\sin(\gamma).$$

We first consider a finite lattice (and hence, a finite operator and matrix), but we are ultimately interested in the behavior of large systems. Accordingly, we will proceed to consider the half-infinite matrix where the bound state(s), if any, of the system will appear, along with the other scattering states.

Because of the asymmetry generated by this perturbation, we employ the transfer-matrix method to find an eigenpair (eigenvalue and eigenvector) for this matrix. We first define the recursive relations that must hold for any eigenvector of H_Λ; these are relatively simple given the tridiagonal nature of the matrix. We express these relations in matrix form and diagonalize this matrix. We then use the diagonalization to find the desired eigenvalue and, from there, proceed to find the eigenvector.

Our results are as follows:

Theorem 3.2. *There exists an eigenpair of H_Λ given by the eigenvalue $\lambda = a - \frac{1}{b-a}$ with the corresponding eigenvector $v = (v_i)_{i=1}^N$, where*

$$v_i = \left(-\frac{1}{b-a}\right)^{i-1}. \tag{3.4}$$

Proof. Denote the eigenvector we are seeking by v, with the x'th element denoted by $v(x)$. Then, the following eigenvector relation must hold for each x:

$$[(H_\Lambda - \lambda\mathbb{I})v](x) = 0. \tag{3.5}$$

To find the eigenvalues, we exploit the tridiagonal nature of H_Λ, which creates a series of simple recursive relations that must hold for any eigenvector of H_Λ, with exceptions only for the first and last elements:

$$\begin{aligned}
\lambda v(1) &= av(1) + v(2)\\
\lambda v(2) &= v(1) + bv(2) + v(3)\\
\lambda v(3) &= v(2) + bv(3) + v(4)\\
&\vdots\\
\lambda v(K) &= v(K-1) + bv(K) + v(K+1)\\
&\vdots\\
\lambda v(N-1) &= v(n-2) + bv(N-1) + v(n)\\
\lambda v(N) &= v(n-1) + bv(N).
\end{aligned}$$

Without loss of generality, we take the first entry of the eigenvector to be unity, i.e., we assume $v(1) = 1$. It then follows from the above equations that $v(2) = \lambda - a$, $v(3) = (\lambda - b)(\lambda - a) - 1$, and so on.

We express the above equations in matrix form as follows:

$$\begin{pmatrix} v(x) \\ v(x+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & \lambda - b \end{pmatrix} \begin{pmatrix} v(x-1) \\ v(x) \end{pmatrix}.$$

Denoting this 2 x 2 transition matrix by T, we find that, for any x,

$$\begin{pmatrix} v(x) \\ v(x+1) \end{pmatrix} = T^{x-1} \begin{pmatrix} 1 \\ \lambda - a \end{pmatrix}. \tag{3.6}$$

At the end of the matrix, we therefore have the following boundary condition that the eigenvalue will need to satisfy:

$$\begin{pmatrix} v(N-1) \\ v(N) \end{pmatrix} = T^{N-2} \begin{pmatrix} 1 \\ \lambda - a \end{pmatrix}. \tag{3.7}$$

We diagonalize the matrix T:

$$T = \begin{pmatrix} \frac{(\lambda-b)+\psi}{2} & \frac{(\lambda-b)-\psi}{2} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{(\lambda-b)-\psi}{2} & 0 \\ 0 & \frac{(\lambda-b)+\psi}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{\psi} & \frac{-(\lambda-b)+\psi}{2\psi} \\ -\frac{1}{\psi} & \frac{(\lambda-b)+\psi}{2\psi} \end{pmatrix}$$

where

$$\psi = \sqrt{(b-\lambda)^2 - 4}.$$

This yields the two eigenvalues

$$\mu_1 = \frac{\lambda - b - \sqrt{(b-\lambda)^2 - 4}}{2}, \mu_2 = \frac{\lambda - b + \sqrt{(b-\lambda)^2 - 4}}{2}. \tag{3.8}$$

We observe that $|\mu_1| < |\mu_2|$.

Next, we can write $\begin{pmatrix} 1 \\ \lambda - a \end{pmatrix}$ in the orthonormal basis given by the columns of V above:

$$\begin{pmatrix} 1 \\ \lambda - a \end{pmatrix} = c_+ \omega_+ + c_- \omega_-$$

so

$$T^k \begin{pmatrix} 1 \\ \lambda - a \end{pmatrix} = c_+ \mu_+^k \omega_+ + c_- \mu_-^k \omega_-.$$

If λ is an eigenvalue of S, then, there must exist C such that

$$\begin{pmatrix} \frac{\lambda - b + \psi(\lambda)}{2} \\ 1 \end{pmatrix} = c \begin{pmatrix} 1 \\ \lambda - a \end{pmatrix}.$$

It immediately follows that

$$c = \frac{1}{\lambda - a}$$

so we have the relation

$$\frac{\lambda - b + \psi(\lambda)}{2} = \frac{1}{\lambda - a}.$$

Solving for λ immediately yields the desired eigenvalue:

$$\lambda = a - \frac{1}{b - a}. \tag{3.9}$$

We can now amend our T matrix equation as follows:

$$\begin{pmatrix} v(x) \\ v(x+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & a-b-\frac{1}{a-b} \end{pmatrix}^{x-1} \begin{pmatrix} 1 \\ \lambda - a \end{pmatrix}.$$

It remains to be shown that the corresponding eigenvector is as given. This is done via induction. For the base cases, we first assume without loss of generality that $v_1 = 1$. It immediately follows from the recursive relations given above that $v_2 = -\frac{1}{b-a}$.

Now, assume that

$$v_{i-2} = \left(-\frac{1}{b-a}\right)^{i-3}$$
$$v_{i-1} = \left(-\frac{1}{b-a}\right)^{i-2}.$$

Then, by direct calculation, using the second equation of the T matrix:

$$\begin{aligned} v_i &= \left(a-b-\frac{1}{a-b}\right)\left(-\frac{1}{a-b}\right)^{i-2} - \left(-\frac{1}{a-b}\right)^{i-3} \\ &= -(b-a)\left(-\frac{1}{a-b}\right)^{i-2} + \left(-\frac{1}{a-b}\right)^{i-1} - \left(-\frac{1}{a-b}\right)^{i-3} \\ &= \left(-\frac{1}{a-b}\right)^{i-3} + \left(-\frac{1}{a-b}\right)^{i-1} - \left(-\frac{1}{a-b}\right)^{i-3} \\ &= \left(-\frac{1}{a-b}\right)^{i-1}, \end{aligned}$$

as desired.

□

3.3 Extension of the Finite Edge Perturbation to an Infinite Lattice

For large lattices, as would be encountered for physical systems where we have, for example, a crystal with a number of atoms of order 10^{23}, it is convenient to consider infinite lattices. In this case, for a one-dimensional chain, the boundary condition Equation 3.7 disappears, and we can use the ℓ^2-norm convergence criterion to determine whether bound states exist. Extending this edge-perturbation case to an infinite lattice is therefore straightforward:

Corollary 3.1. *The eigenvalues and eigenvectors of H_Λ corresponding to the finite Λ in Figure 2 are precisely identical to those of the Hamiltonian H_Γ corresponding to the half-infinite extension of that lattice.*

Proof. The exact same steps can applied for the infinite case as were applied for the finite case, with the exception that one of the terminal conditions (that for the bottom of the matrix) can now be discarded, since there is no "end" to the matrix. Accordingly, the eigenpairs remain the same. □

Since the matrix is now infinite, different eigenstates now fall into two categories:

1. **Continuous spectrum values**, with corresponding scattering (or delocalized) states. These satisfy the equation

$$H\psi = \lambda\psi, \psi \in \ell^\infty(\Gamma) \text{ but } \psi \notin \ell^2(\Gamma).$$

2. **Bound state eigenvalues**, with corresponding localized eigenvectors. These satisfy the equation

$$H\psi = \lambda\psi, \psi \in \ell^2(\Gamma).$$

An vector ψ in an ℓ^2 space is square-summable, meaning that

$$||\psi||_2^2 = \sum_{i=1}^{\infty} |\psi_i|^2 < \infty.$$

On the other hand, if ψ is in ℓ^∞ space, but not ℓ^2 space, it is not square-summable. This has an important physical interpretation: For a normalized wavevector ψ, the probability that the particle is found at location x is given by ψ_x [2]:

$$P(\text{particle is at } x) = \frac{|\psi_x|^2}{||\psi||_2^2}$$

A wavevector that is not square-summable, however, cannot be normalized.[2]

We also know that the existence of a bound state due to the addition of a rank-one perturbation to the operator H will not change the continuous spectrum values of the original operator. This is guaranteed by Weyl's Theorem [1].

As an example, the eigenvector we found in Theorem 3.2 above gives us a bound state for H_Γ, under certain conditions:

Theorem 3.3. *A bound state for H_Γ as defined above exists exactly when $b \notin [a-1, a+1]$.*

[2]Intuitively, since $||\psi||_2^2 = \infty$,

$$P(\text{particle is at } x) = \frac{|\psi_x|^2}{||\psi||_2^2} = \frac{(\text{finite number})}{\infty} \text{“}=\text{”}\ 0.$$

Hence, there is a zero probability of finding the particle in any given location!

Proof. As discussed, a bound state exists precisely when the wavevector ψ corresponding to that state is square-summable, i.e., $\psi \in \ell^2(\Gamma)$. Here, square-summability is easily verified:

$$\begin{aligned}\sum_{i=1}^{\infty} v_i &= \sum_{i=1}^{\infty} \left(-\frac{1}{b-a}\right)^{i-1} \\ &= \frac{1}{1-\left(-\frac{1}{b-a}\right)} \\ &= \frac{1}{1+\left(\frac{1}{b-a}\right)} < \infty\end{aligned}$$

exactly when

$$-1 < \frac{1}{b-a} < 1.$$

Then, we have two cases:

- **Case 1:** If $b > a$, then $b - a > 0 \Rightarrow -(b-a) < 1 < b - a \Rightarrow b > a + 1$.
- **Case 2:** If $b < a$, then $b - a < 0 \Rightarrow -(b-a) > 1 > b - a \Rightarrow b < a + 1$.

Hence, b cannot be in the interval $[a-1, a+1]$. □

We can also observe that the continuous spectrum of the Hamiltonian operator remains invariant under the perturbation (see Figure 6), in accordance with Weyl's Theorem.

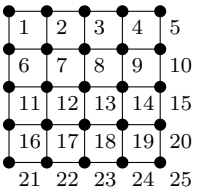
4 Finite Two-dimensional Lattices

At this point, we have considered the main basic cases in one dimension (finite and infinite lattices with and without perturbations), and it is natural to now consider analogous lattices in two dimensions.

Our above method from Section 2 generalizes immediately to the two-dimensional case, where the top and bottom boundaries are joined, as well as the left and right edges. For a lattice with n rows of sites and m columns, we label the lattice by counting from left to right and from the top down (see Figure 3 for an example). To be mathematically precise, for each lattice site, we count the number of rows before the row containing the site, and the column wherein the site is located. For example, the lattice site on the 3^{rd} row, 5^{th} column would be labeled site number $2m + 5$.

Using this system of indexing, we can find the matrix representation of the Hamiltonian. Again, the matrix is circulant; therefore, the same technique with a Discrete Fourier Transform matrix can be applied:

Figure 3: A finite two-dimensional lattice with sites labeled



Theorem 4.1. *The eigenvalues for the matrix H above are*

$$\lambda_t = 4 - 2\cos\left(\frac{2\pi i}{M}t\right) - 2\cos\left(\frac{2\pi i}{M}mt\right), \tag{4.1}$$

with associated eigenvectors

$$f_t = (1, \omega^t, \omega^{2t}, \ldots, \omega^{(mn-1)t}) \ \ \forall t = 1, \ldots, mn. \tag{4.2}$$

for $i = 1, \ldots, mn$.

Proof. As before, columns of the DFT matrix are the eigenvectors of H, as shown above. The eigenvalues follow:

$$\begin{aligned}(\lambda_t f_t)_s &= (A_t f_t)_s \\ &= 4(f_t)_s - (f_t)_{s-1} - (f_t)_{s+1} - (f_t)_{s-m} - (f_t)_{s+m} \\ &= 4\omega^{(s-1)t} - \omega^{(s-2)t} - \omega^{st} - \omega^{(s-m-1)t} - \omega^{(s-m+1)t}\end{aligned}$$

Additionally, using the eigenvectors explicitly:

$$\begin{aligned}(\lambda_t f_t)_s &= \lambda_t (f_t)_s \\ &= \lambda_t \omega^{(s-1)t}\end{aligned}$$

Setting the two expressions equal, we have our eigenvalues:

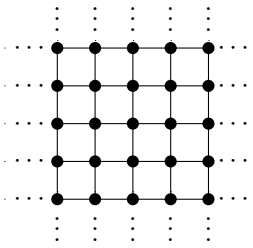
$$\begin{aligned}\lambda_t &= 4 - \omega^{-t} - \omega^t - \omega^{-mt} - \omega^{mt} \\ &= 4 - \cos\left(-\frac{2\pi i}{M}t\right) - i\sin\left(-\frac{2\pi i}{M}t\right) - \cos\left(\frac{2\pi i}{M}t\right) - i\sin\left(\frac{2\pi i}{M}t\right) \\ &\quad - \cos\left(-\frac{2\pi i}{M}mt\right) - i\sin\left(-\frac{2\pi i}{M}mt\right) - \cos\left(\frac{2\pi i}{M}mt\right) - i\sin\left(\frac{2\pi i}{M}mt\right) \\ &= 4 - 2\cos\left(\frac{2\pi i}{M}t\right) - 2\cos\left(\frac{2\pi i}{M}mt\right) \ \ \forall j = 1, \ldots, M,\end{aligned}$$

as desired. □

5 The Entire Infinite Two-dimensional Lattice

Our results above naturally generalize to two dimensions when the entire two-dimensional lattice (Figure 4) is considered.

Figure 4: The entire infinite two-dimensional lattice



In the one-dimensional case, we found that

$$(H_\Lambda^{(1)}\psi)_x = \psi_x + \psi_{x+1} + c\psi_x$$

which, in terms of the basis vectors of the Hilbert space, implies that

$$\begin{aligned}(H_\Lambda^{(1)}e_z)_x &= \delta_{x-1} + \delta_{x-1} + c\delta_x \\ &= e_{z+1} + e_{z-1} + ce_z.\end{aligned} \tag{5.1}$$

The two-dimensional case is analogous; for any lattice site (x, y), and denoting $e_{(x,y)} = e_x \otimes e_y$, the Hamiltonian is given by

$$\begin{aligned}H^{(2)}e_{(x,y)} &= H^{(2)}(e_x \otimes e_y) \\ &= e_{x-1} \otimes e_y + e_{x+1} \otimes e_y + e_x \otimes e_{y-1} + e_x \otimes e_{y+1} + ce_x \otimes e_y \\ &= H^{(1)}e_x \otimes e_y - ce_x \otimes e_y + e_x \otimes H^{(1)}e_y - ce_x \otimes e_y + ce_x \otimes e_y\end{aligned}$$

We can therefore factor this expression using the tensor product:

$$\begin{aligned}(H^{(2)} - c\mathbb{1} \otimes \mathbb{1})(e_x \otimes e_y) &= [(H^{(1)} - c\mathbb{1}) \otimes \mathbb{1} + \mathbb{1} \otimes (H^{(1)} - c\mathbb{1})](e_x \otimes e_y) \\ (H^{(2)} - c\mathbb{1} \otimes \mathbb{1}) &= [(H^{(1)} - c\mathbb{1}) \otimes \mathbb{1} + \mathbb{1} \otimes (H^{(1)} - c\mathbb{1})]\end{aligned} \tag{5.2}$$

The eigenvalues of $H^{(2)}$ are easily found as a result:

Theorem 5.1. *The eigenvectors for the matrix $H^{(2)}$, corresponding to the discrete second-order Laplacian on the entire two-dimensional lattice, are*

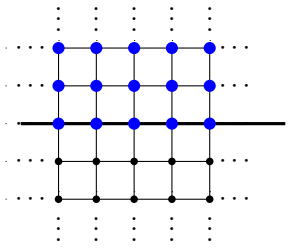
$$w_{kl} = v_k \otimes v_l \tag{5.3}$$

with corresponding eigenvalues

$$\eta_{kl} = \lambda_k + \lambda_l - c, \tag{5.4}$$

where λ_k is the eigenvalue corresponding to the eigenvector v_k of the one-dimensional matrix $H^{(1)}$ and c is defined in Equation (5.1).

Figure 5: Half-infinite lattice with boundary slope of 0



Proof. This follows directly from our computations above:

$$\begin{aligned} H^{(2)}w_{kl} &= (c\mathbb{1}\otimes\mathbb{1})w_{kl} + (H^{(1)} - c\mathbb{1})\otimes\mathbb{1}w_{kl} + \mathbb{1}\otimes(H^{(1)} - c\mathbb{1})w_{kl} \\ &= cw_{kl} + (H^{(1)} - c\mathbb{1})v_k \otimes v_l + v_k \otimes (H^{(1)} - c\mathbb{1})v_l \\ &= cw_{kl} + (\lambda_k - c)v_k \otimes v_l + v_k \otimes (\lambda_l - c)v_l \\ &= cw_{kl} + (\lambda_k - c)w_{kl} + (\lambda_l - c)w_{kl} \\ &= (\lambda_k + \lambda_l - c)w_{kl}, \end{aligned}$$

as desired. □

6 Existence of the Zero Ground State

Next, we wish to begin considering half-infinite lattices with differing types of boundaries. Another author considers a boundary of slope 0, with respect to the x-axis, as shown in Figure 5 [4].

We shall extend this investigation to consider two similar, but distinct, two-dimensional half-infinite lattices in this paper, with boundary slopes of 1 and 2. These correspond to boundary angles of $\arctan(1) = \pi/4 = 45^\circ$ and $\arctan(2) \approx 1.11 \approx 63.43^\circ$ (relative to the x-axis), respectively.

Before we do so, however, it is useful to state and prove a general theorem guaranteeing the existence of a zero eigenvalue for the cases we consider.

First, we generalize the second-order discrete Laplacian, the Hamiltonian operator we have been using up until this point, by allowing two parameters (those quantifying nearest-neighbor interactions) to vary:

Definition 6.1. *Define the generalized second-order discrete Laplacian Hamiltonian $\tilde{H}_\Lambda$ by*

$$\tilde{H}_\Lambda = \sum_{x \twoheadrightarrow y} h_{xy},$$

where the summation is over all directed edges $x \twoheadrightarrow y$ for $x, y \in \Lambda$, and

$$h_{xy} = \begin{cases} |\lambda_h e_x - e_y\rangle \langle \lambda_h e_x - e_y| & \text{if } x \twoheadrightarrow y \text{ is horizontal} \\ |\lambda_v e_x - e_y\rangle \langle \lambda_v e_x - e_y| & \text{if } x \twoheadrightarrow y \text{ is vertical} \end{cases}$$

for $\lambda_h, \lambda_v \geq 0$.

This definition of the Hamiltonian $\tilde{H}$ subsumes the Hamiltonian H previously defined in Definition 1.1 as the special case in which $\lambda_h = \lambda_v = 1$. Accordingly, from this point forward, the operator H shall refer to that defined in Definition 6.1. In addition, all Hamiltonians of this class are positive semidefinite, as defined as follows:

Definition 6.2. *A matrix is positive semidefinite (denoted* $A \geq 0$*) if* $A^* = A$ *and* $\langle \psi, A\psi \rangle \geq 0$ *for all vectors* ψ.

A very useful property of positive semidefinite matrices is that all of its eigenvalues are non-negative. In addition, the following Lemma provides another useful property:

Lemma 6.1. *For positive semidefinite matrices* $A_i \geq 0, i = 1, ..., N$,

$$\left(\sum_{i=1}^{N} A_i \right) \psi = 0$$

if and only if $A_i \psi = 0$ *for all* $i = 1, ..., N$.

Proof. Suppose A_1, A_2 are positive semidefinite matrices and $(A_1 + A_2)\psi = 0$. Then, $\psi^T (A_1 + A_2)\psi = 0$. Because A_1, A_2 are positive semidefinite, we know that $\psi^T A_1 \psi, \psi^T A_2 \psi \geq 0$, so it must follow that $\psi^T A_1 \psi = \psi^T A_2 \psi = 0$. Because A_1 is positive semidefinite, there exists B such that $A_1 = B^T B$. Then, $\psi^T B^T B \psi = (B\psi)^T B\psi = 0$, which implies that $B\psi = 0$ (since a self-orthogonal vector is a zero vector, by definition). But then $A_1 \psi = B^T B \psi = 0$. The case for $A_2 \psi$ is analogous. The generalization to an arbitrary number of matrices A_i follows by induction, and the reverse direction of the lemma is trivial. □

These properties help us prove the main theorem of this section:

Theorem 6.1. *For any finite, connected lattice such that* $\lambda_h, \lambda_v > 0$*, there will exist a unique ground state (i.e., there will exist exactly one* ψ *such that* $H_\Lambda \psi = 0$*), up to normalization.*

Proof. By the lemma above, $H_\Lambda \psi = \left(\sum_{x \twoheadrightarrow y} h_{xy} \right) \psi = 0$ if and only if $h_{xy} = 0$ for all directed $x \twoheadrightarrow y$ of the lattice Λ. It thus suffices to show that there exists a unique solution ψ, up to normalization, of the equations $h_{xy} = 0$, i.e.,

$$\begin{aligned} h_{xy} |\psi\rangle &= \lambda^2 |e_x\rangle \langle e_x | \psi_x e_x\rangle + |e_y\rangle \langle e_y | \psi_x e_x\rangle \\ &\quad - \lambda |e_x\rangle \langle e_y | \psi_y e_y\rangle - \lambda |e_y\rangle \langle e_x | \psi_x e_x\rangle \\ &= \psi_x \lambda^2 |e_x\rangle + \psi_y |e_y\rangle - \lambda \psi_y |e_x\rangle - \lambda \psi_x |e_y\rangle \\ &= 0, \end{aligned} \tag{6.1}$$

where λ represents either λ_v or λ_h, whichever applies, depending upon whether x and y are joined horizontally or vertically. Express ψ in the canonical basis:

$$\psi = \sum_{x=1}^{|\Lambda|} \psi_x \left| e_x \right\rangle$$

Then, for any $x \twoheadrightarrow y$, $h_{xy} = 0$ if and only if $\lambda\psi_x - \psi_y = 0$, or

$$\lambda\psi_x = \psi_y. \tag{6.2}$$

(Again, here λ represents either λ_v or λ_h, as appropriate.)

Now, we assumed that Λ is a connected lattice. Hence, if the set of equations given by Equation 6.2 is consistent for all edges, then fixing one value ψ_1 will uniquely determine all other components of ψ.

To see that consistency indeed holds, we observe that Equation 6.2 simply states that, when moving one lattice site up (resp. to the right), the value of the corresponding component of ψ will be multiplied by a factor of $1/\lambda_v$ (resp. $1/\lambda_h$). Since λ_v, λ_h do not depend upon the lattice sites x or y involved, the successive multiplications and divisions imply that the resulting relation between the components corresponding to any two lattice sites $z_1, z_2 \in \Lambda$ will be independent of the path taken between them.

□

7 A Conjecture on the Relationship between Bound States and Spectral Gaps

Let us briefly revisit the model we considered in Section 3.2, in which we perturbed a one-dimensional free-end chain with a varying constant $c = b - a$. According to Proposition 3.3, this results in a bound state exactly when the absolute value of the perturbation is greater than 1. We can also examine the spectral properties resulting from this perturbation. Figure 6 shows the eigenvalues of the perturbed Hamiltonian as functions of the value of the perturbation c. Observe that at $c = 1$, the ground state eigenvector begins to separate from the rest of the continuous spectrum, resulting in a spectral gap.

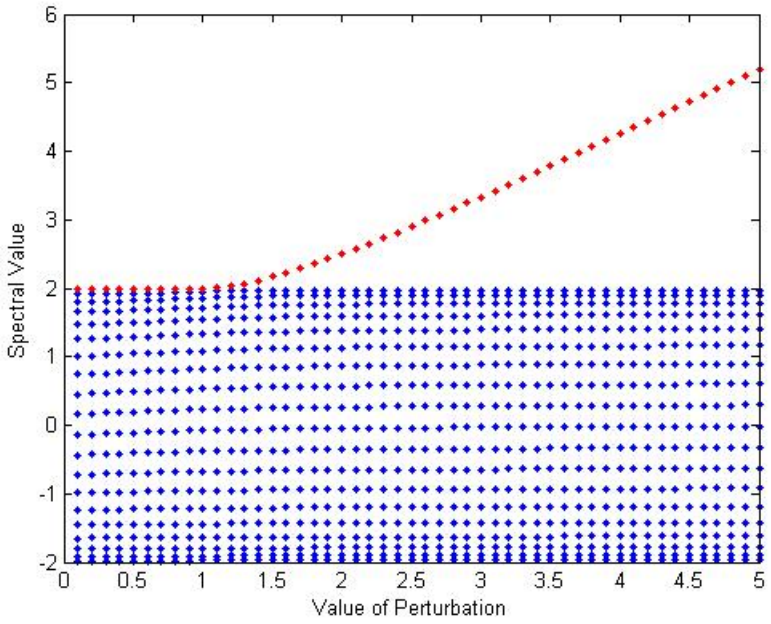
We now consider whether these two phenomena—the beginning of a spectral gap and the beginning of the appearance of a bound state—are related to each other. Formally, we wish to test the following:

Conjecture. *The conditions under which the spectral gap appears are also the conditions under which bound states exist.*

To test this conjecture, we consider the two aforementioned different half-infinite lattices, each with a different boundary:

1. **Boundary slope 1:** This lattice, depicted in Figure 7a, has a boundary angle of $\arcsin(1) = 45°$.

Figure 6: Bound state of the model in Section 3.2 when perturbed



2. **Boundary slope 2:** This lattice, depicted in Figure 7b, has a boundary angle of $\arcsin(2) \approx 63^\circ$.

We proceed with analyzing the first lattice in close detail.

8 The Half-infinite Two-dimensional Lattice with Boundary Slope 1

To model the behavior of an infinite two-dimensional half-infinite lattice with a boundary slope of 1, we construct triangle lattices with such a boundary, with the points labeled as in Figure 8.

The total number of points for such a lattice Λ of n columns and rows is given by the n-th triangular number, i.e.,

$$|\Lambda| = \binom{n}{2}. \tag{8.1}$$

For example, for the lattice in Figure 8, the total number of lattice points is

$$|\Lambda| = \binom{5}{2} = 20.$$

Figure 7: Half-infinite lattices with boundary slopes of 1 and 2

(a) Boundary slope 1 (b) Boundary slope 2

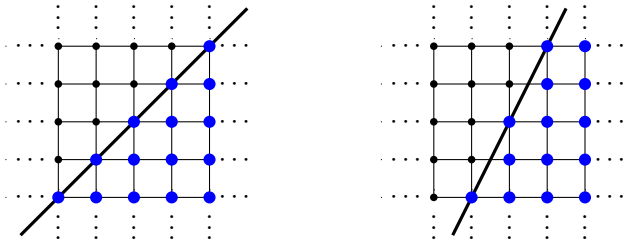
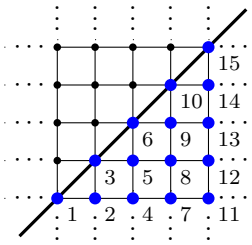


Figure 8: Half-infinite lattice with boundary slope 1 with sites labeled



For sake of simplicity, the remainder of this paper shall refer to this n (the number of columns in the lattice) as the lattice size instead of the actual number of points in the lattice.

In this case, instead of considering perturbations, we shall be interested in when and how the two parameters λ_v and λ_h of the generalized Laplacian operator affect the existence of a bound state or a spectral gap for that operator.

To test our conjecture, there are two properties of any Hamiltonian pertaining to this lattice that we must know: the spectral gap, and the existence or absence of a bound state.

The spectral gap of a matrix H is defined as the difference between the two smallest eigenvalues of H. (Recall that, for our purposes, H is always Hermitian, and hence all its eigenvalues are real; as a result, "small" makes sense in this context.) According to Theorem 6.1, the smallest eigenvalue for the cases we consider here is always 0; hence, the spectral gap is simply the second smallest eigenvalue.

Determining whether the ground state is also a bound state, however, is not quite so straightforward. Since we are considering finite lattices as an approximation of their half-infinite counterparts, we will not have any wavevectors that are divergent in the 2-norm (i.e., there will exist no ψ such that $\psi \in \ell^\infty(\Lambda)$ but $\psi \notin \ell^2(\Lambda)$). We will therefore consider the following quantity, which (for lack of a better name) shall be called the "norm ratio", as a function of the ground

state eigenvector $\widetilde{\psi}$:

$$R(\widetilde{\psi}) = \frac{||\widetilde{\psi}||_\infty}{||\widetilde{\psi}||_2}. \tag{8.2}$$

If this quantity approaches 0 as the size of the lattice increases, then the 2-norm grows faster than the ∞-norm. This would appear to indicate that as the lattice becomes larger (and hence, $\widetilde{\psi}$ becomes longer), there will be terms toward the end of the vector sufficiently large to overpower the previous terms, suggesting divergence of $\widetilde{\psi}$ in the 2-norm. If true, this would correspond to a scattering state.

On the other hand, if this quantity approaches some constant $\xi \neq 0$ as the size of the lattice increases, then it would appear that the larger terms in $\widetilde{\psi}$ appear toward the beginning of the vector, which would suggest convergence of the 2-norm and, thus, membership in the ℓ^2 space, and would correspond to a bound state.

Although it is not difficult to construct a matrix representation of the Hamiltonian for this lattice (or, for that matter, any other lattices considered in this paper), it is not immediately obvious how such matrices could be diagonalized analytically (e.g., by the methods considered in previous sections). However, both of the key quantities just described—the spectral gap and the norm ratio—can be readily calculated numerically by MATLAB. Accordingly, our analysis from this point on will rely heavily upon such computations. (Code for all of the MATLAB routines necessary to replicate these results is given in the Appendix.)

As our first step, we need to determine how large of a lattice we need to use in our computations. To do this, we plot the norm ratio $R(\widetilde{\psi})$ against n (the number of columns in the lattice) for various values of λ_v and λ_h. This is shown in Figure 9. We look for a value of n at which point we can reasonably expect to be able to differentiate between bound state vectors and continuous spectrum vectors. From this figure, $n = 30$ appears to be a reasonable choice.

Next, we examine the spectral properties and probable bound states of the Hamiltonian for various values of λ_h and λ_v. These are also best examined with plots. Figure 10 gives the norm ratios, and Figure 11 gives the spectral gaps, of Hamiltonians with varying parameter values.

These figures together show that for the λ_h, λ_v values attempted, the minimum spectral gap is indeed the location at which the norm ratio is the smallest.

With this information, it is possible to classify, for each (λ_h, λ_v) pair, whether the particle will tend to be localized in a bound state, and furthermore, how it will tend to be localized. From experimental data, it appears that the particle has seven different general types of behaviors, as illustrated in Figures 12 through 18, depending upon the values of λ_h, λ_v:

1. The particle is localized on the diagonal boundary (Figure 12).
2. The particle is localized on the upper right corner (Figure 13).
3. The particle is localized on the right boundary (Figure 14).
4. The particle is localized on the lower right corner (Figure 15).

Figure 9: Norm ratio by n for various values of λ_h and λ_v (slope = 1)

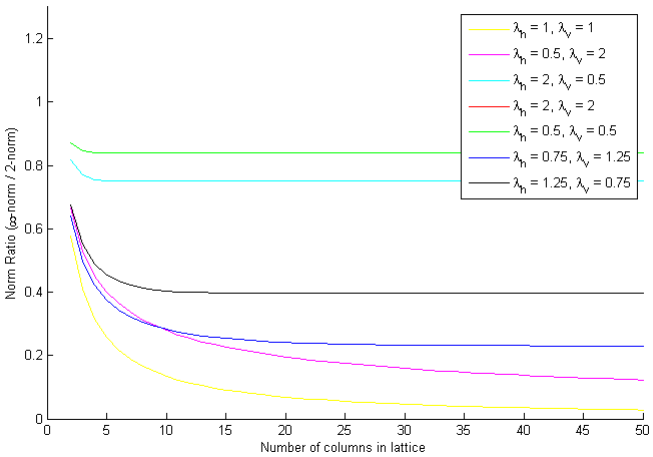
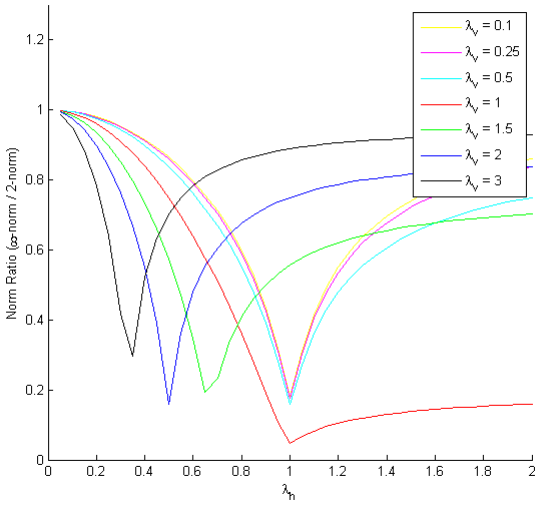


Figure 10: Norm ratios for various values of λ_h and λ_v (size = 30, slope = 1)



5. The particle is localized on the lower boundary (Figure 16).

6. The particle is localized on the lower left corner (Figure 17).

7. The particle does not localize and remains in a scattered state (Figure 18).

For an intuitive view of how the λ_h, λ_v parameters affect the behavior of the particle, Figure 19 classifies the behavior of the particle at values of λ_h, λ_v between 0.1 and 2 based upon visual inspection. The seven types of behaviors listed above correspond to the following symbols, respectively:

1. Yellow "x" denotes localization on the diagonal boundary.

2. Blue square denotes localization on the upper right corner.

3. Pink star denotes localization on the right boundary.

4. Red triangle denotes localization on the lower right corner.

5. Cyan asterisk denotes localization on the lower boundary.

6. Green circles denotes localization on the lower left corner.

7. Black diamond denotes a delocalized (scattering) state.

9 The Half-infinite Two-dimensional Lattice with Boundary Slope 2

Our analysis for a lattice of slope 2 is similar. The number of points in a lattice of this type with n columns is

$$|\Lambda| = \sum_{i=1}^{n}(2i-1) = 2\sum_{i=1}^{n}(i) - n = 2\left(\frac{n(n+1)}{2}\right) - n = n^2. \tag{9.1}$$

Again, from Figure 20, we see that a lattice of $n = 30$ columns should be sufficient for our purposes.

Next, like before, we examine the spectral properties and probable bound states of the Hamiltonian for various values of λ_h and λ_v. Figure 21 gives the norm ratios, and Figure 22 gives the spectral gaps, of Hamiltonians with differing parameter values. Again, we observe that the minimum values of both the norm ratios and the spectral gaps appear to coincide in each case.

10 Conclusions

In this paper, we first considered several simple cases in one and two dimensions, which we solved with several different methods (Discrete Fourier Transform, transfer matrices, and *Ansatz*). These results motivated the main conjecture

Figure 11: Spectral gaps for various values of λ_h and λ_v (size = 30, slope = 1)

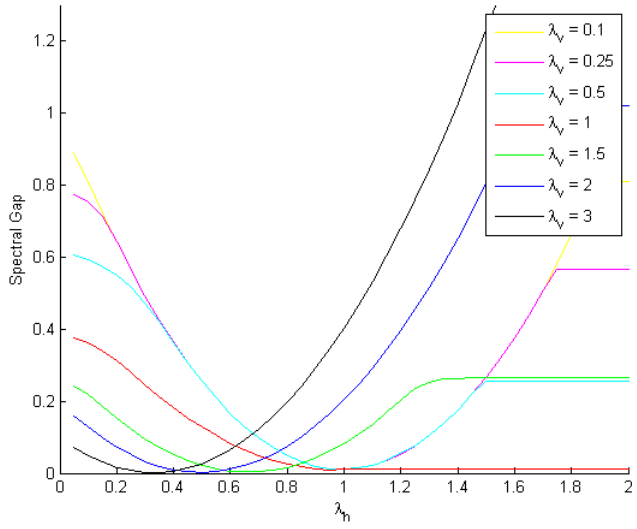


Figure 12: Density plot for $\lambda_v = 1.7, \lambda_h = 0.6$ (size = 30, slope = 1)

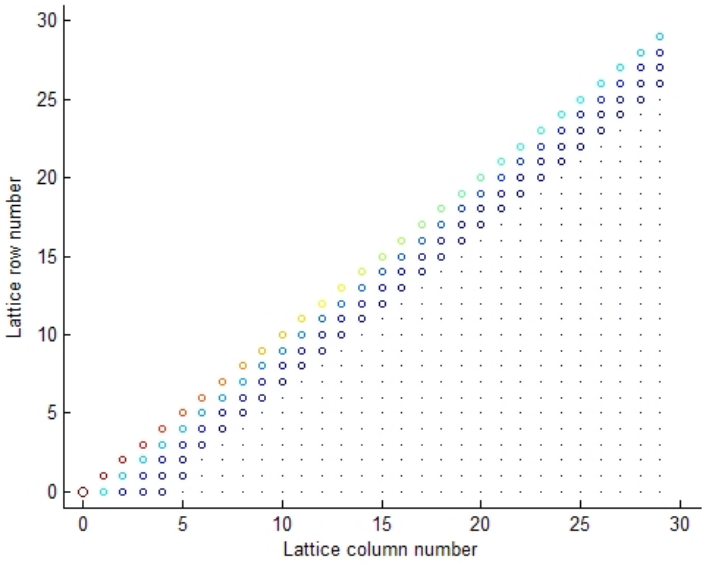


Figure 13: Density plot for $\lambda_v = 0.6, \lambda_h = 0.6$ (size = 30, slope = 1)

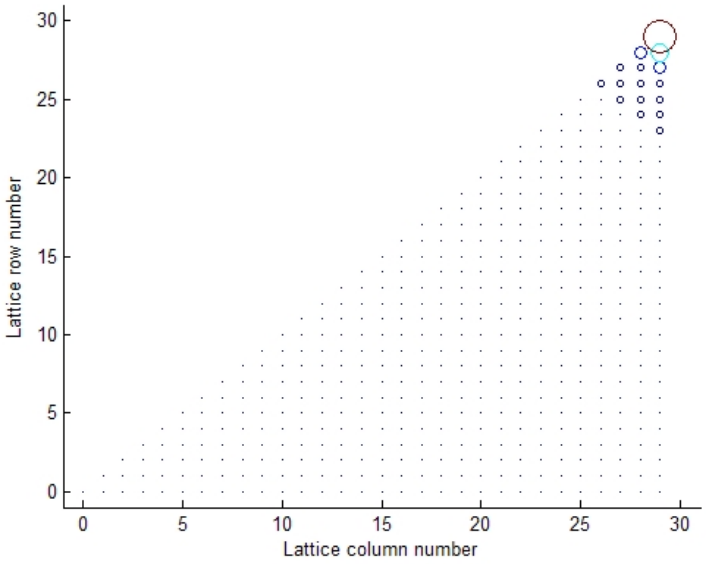


Figure 14: Density plot for $\lambda_v = 0.4, \lambda_h = 1.0$ (size = 30, slope = 1)

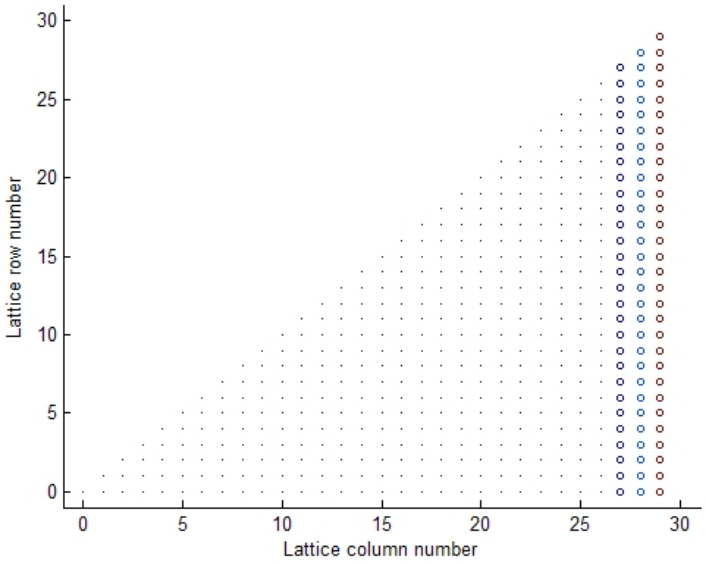


Figure 15: Density plot for $\lambda_v = 0.6, \lambda_h = 1.6$ (size = 30, slope = 1)

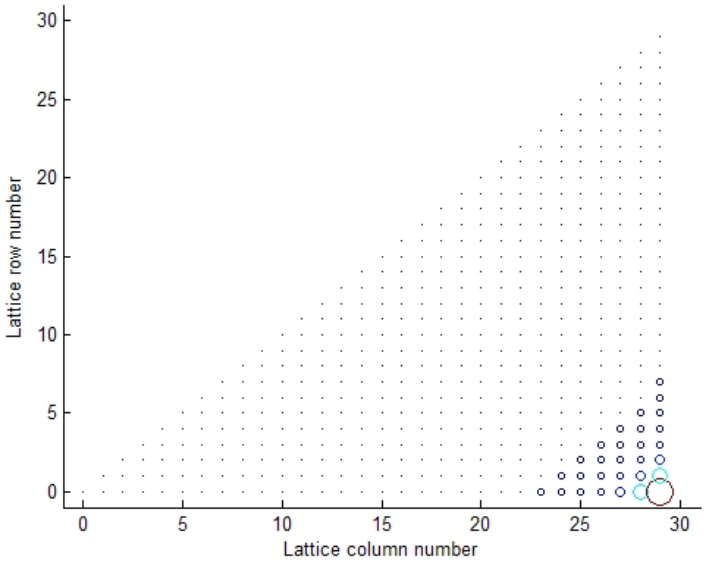


Figure 16: Density plot for $\lambda_v = 1.0, \lambda_h = 1.6$ (size = 30, slope = 1)

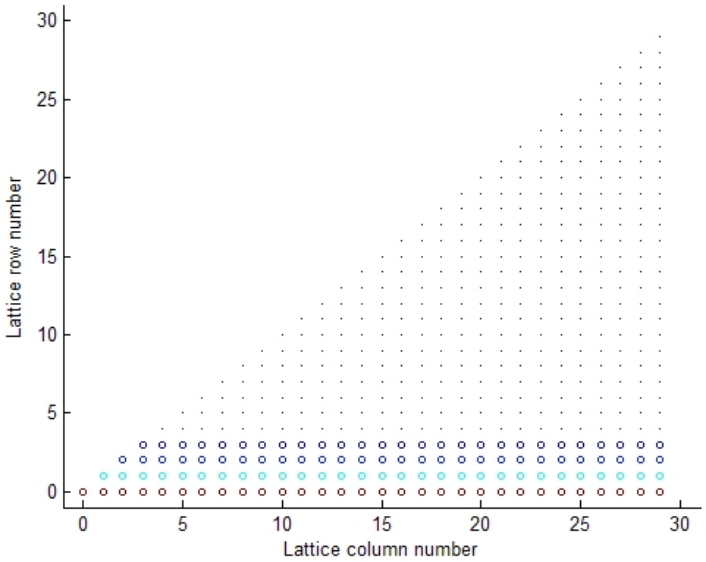


Figure 17: Density plot for $\lambda_v = 1.6, \lambda_h = 1.6$ (size = 30, slope = 1)

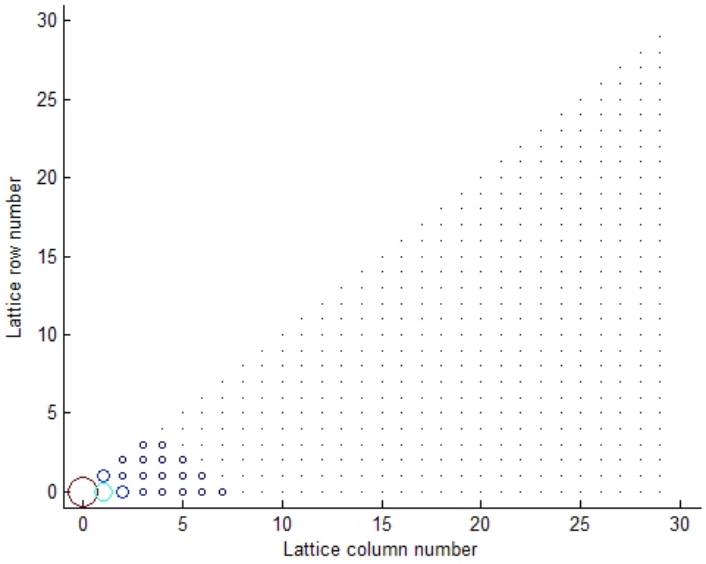


Figure 18: Density plot for $\lambda_v = 1.0, \lambda_h = 1.0$ (size = 30, slope = 1)

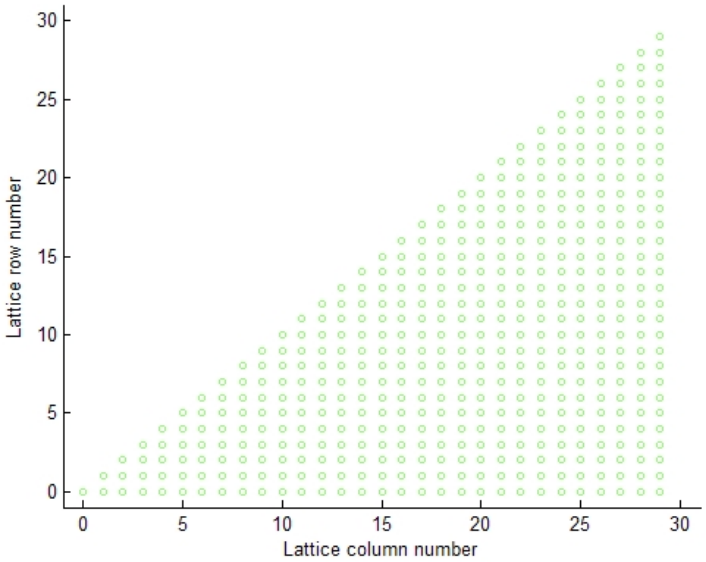


Figure 19: Phase diagram of particle states by λ_v, λ_h (size = 30, slope = 1)

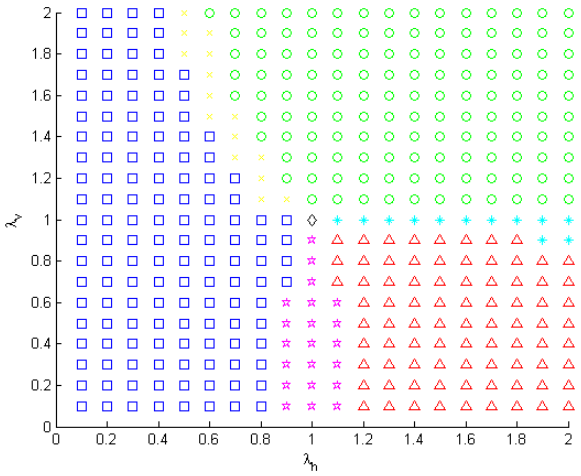


Figure 20: Norm ratio by n for various values of λ_h and λ_v (slope = 2)

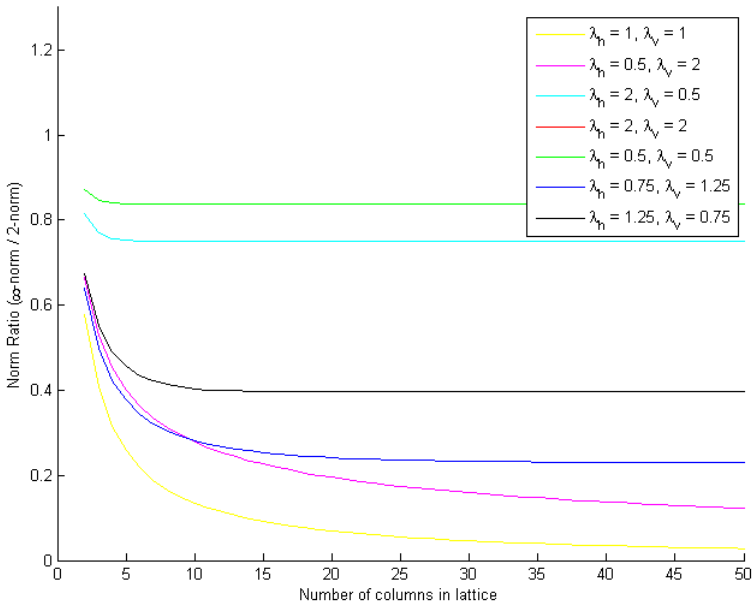


Figure 21: Norm ratios for various values of λ_h and λ_v (size = 30, slope = 2)

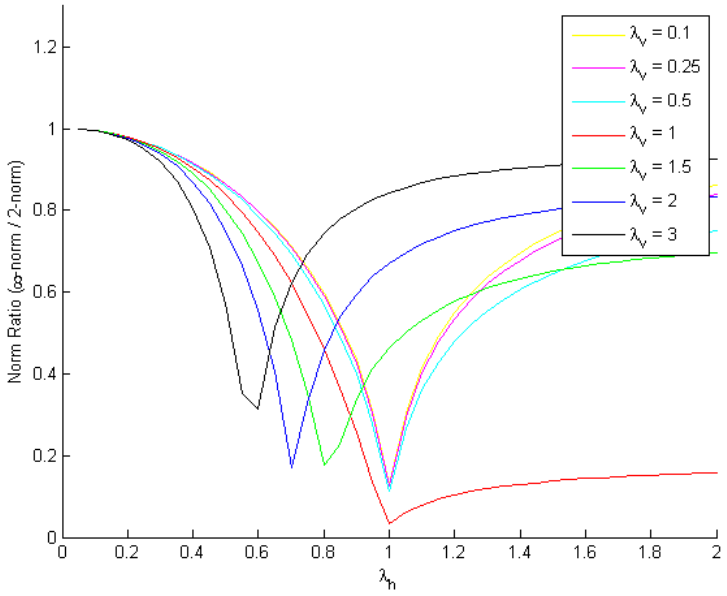
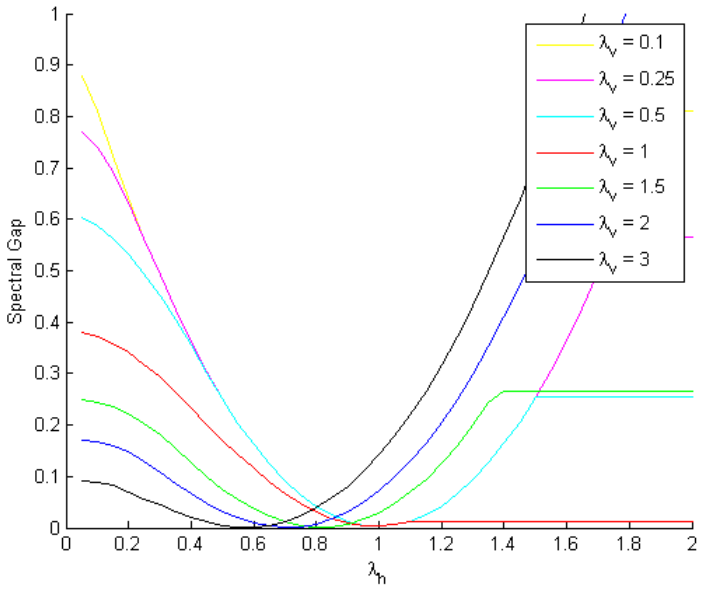


Figure 22: Spectral gaps for various values of λ_h and λ_v (size = 30, slope = 2)



we investigated, which we tested by varying two parameters in the Hamiltonian and observing the resulting spectral gaps and existence of bound states. The two different half-infinite lattices we considered in the latter portion of the paper seem to support the conjecture; in each of the cases, the minimal spectral gap coincides with the probable appearance of a bound state, as evidenced by the ratio of the ∞-norm to the 2-norm of the ground state eigenvector. The analysis also showed the effects of varying the nearest-neighbor interaction parameters upon the spectral properties of the Hamiltonian.

We hope to extend this research in the future by exploring methods of analytically investigating lattices such as the ones we have considered in the two preceding sections. In addition, whereas in this paper we have considered lattices with boundaries of slopes 1 and 2, we hope in the future to be able to construct and analyze lattice boundaries of arbitrary slopes. Finally, we wish to consider other methods of approximating half-infinite lattices such as those considered here. In this paper, we have built successively larger triangles. However, it is uncertain whether this is the most suitable way to construct such lattices, and the different types of behavior of a localized particle may be fewer than the seven presented. This may become more apparent with a different way of constructing these finite lattices.

Acknowledgments

I wish to thank Professor Bruno Nachtergaele for his most generous and patient guidance and support throughout this project. I also would like to express my appreciation for my family—A.H., A.H., A.H., and A.H.—for their support and for bearing with a often very preoccupied Aaron throughout this project, as well as C.H., J.H, J.L., M.P., and E.Y., and my fellow classmates at SIBS-NCSU 2012 (especially C.G. and J.Y.), for their continued encouragement of my mathematical endeavors. Finally, I would like to acknowledge all of the wonderful professors, instructors, and classmates I have had the opportunity to learn from during my time here at UC Davis.

References

[1] Christian Remling, Functional Analysis: Perturbation by Compact Operators. `http://www2.math.ou.edu/~cremling/teaching/lecturenotes/fa-new/ln15.pdf`

[2] David J. Griffiths, *Introduction to Quantum Mechanics*, Pearson Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2005.

[3] Limeng Feng, Vadim Linetsky, Electronic Companion to "Pricing Options in Jump-Diffusion Models: An Extrapolation Approach", *Operations Research*, 56:2, ec1-ec3 (2007).

[4] Amanda Young, personal communication.

Appendix: MATLAB Code

The analysis in Sections 8 and 9 employed MATLAB extensively to compute eigenvalues, eigenvectors, norms, spectral gaps, and other properties of the Hamiltonian operators considered. This Appendix contains the code for all of the computations performed in those sections.

Note: In the following functions, the `n` parameter specifies the number of points in the lattice, rather than the number of columns like was done in the paper.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       VertexList1.m
%%% Function:   VertexList1(last_n)
%%% Purpose:    Creates a list of all vertex coordinates with
%%% corresponding lattice site numbers up to the last point last_n
%%% for a lattice of slope 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function vertex_list = VertexList1(last_n)

% Initialize variables
old_x = 0;
old_y = 0;
vertex_list = [];

for vertexnum = 1:last_n

    % If next point is below y = x, add the next point up
    if old_y + 1 <= old_x
        new_x = old_x;
        new_y = old_y + 1;

    % If next point is above y = x, add the next point in the next ...
        column
    elseif old_y + 1 > old_x
        new_x = old_x + 1;
        new_y = 0;
    end

    % Record new_x, new_y corresponding to n
    temp = [vertexnum, old_x, old_y]';
    vertex_list = horzcat(vertex_list, temp);

    % Set new variables as old variables to prepare for the next ...
        iteration
    old_y = new_y;
    old_x = new_x;

end

% Uncomment to plot the resulting lattice
% scatter(vertex_list(2,:), vertex_list(3,:))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       VertexList2.m
%%% Function:   VertexList2(last_n)
%%% Purpose:    Creates a list of all vertex coordinates with
%%% corresponding lattice site numbers up to the last point last_n
%%% for a lattice of slope 2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function vertex_list = VertexList2(last_n)

% Initialize variables
old_x = 0;
old_y = 0;
vertex_list = [];

for vertexnum = 1:last_n

    % If next point is below y = 2x, add the next point up
    if old_y + 1 <= 2*old_x
        new_x = old_x;
        new_y = old_y + 1;

    % If next point is above y = 2x, add the next point in the next ...
        column
    elseif old_y + 1 > 2*old_x
        new_x = old_x + 1;
        new_y = 0;
    end

    % Record new_x, new_y corresponding to n
    temp = [vertexnum, old_x, old_y]';
    vertex_list = horzcat(vertex_list, temp);

    % Set new variables as old variables to prepare for the next loop
    old_y = new_y;
    old_x = new_x;

end

% Uncomment to plot the resulting lattice
% scatter(vertex_list(2,:), vertex_list(3,:))



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       Ham1.m
%%% Function:   Ham1(n, lambda_v, lambda_h)
%%% Purpose:    Creates the Hamiltonian operator for a lattice of n
%%% points and slope 1 with nearest-neighbor interaction weights
%%% lambda_v and lambda_h.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function H = Ham1(n, lambda_v, lambda_h)

% Initialize vector of edges
listofhedges = [1;2];
listofvedges = [];
```

```
% Loop through each column of the lattice except the last; this ...
    gives all edges besides vertical edges for outermost column of ...
    lattice
for latticecol = 2:(n-1)

    % Edge for boundary vertex
    boundary = nchoosek(latticecol+1,2);                              ...
        % Calculate the boundary point

        % Edge going to the right
        start = boundary;                                             ...
            % Calculate the starting point
        finish = (boundary + latticecol);                             ...
            % Calculate the ending point
        boundaryedge = [start, finish]';                              ...
            % Put together the edge
        listofhedges(1:2,size(listofhedges,2)+1) = boundaryedge;      ...
            % Append the edge to the list of edges

    % Edges for non-boundary vertices
    for current = (nchoosek(latticecol,2)+1):nchoosek(latticecol+1,2)-1

        % Edge going above
        start = current;
        finish = current+1;
        aboveedge = [start, finish]';
        listofvedges(1:2,size(listofvedges,2)+1) = aboveedge;

        % Edge going to the right
        start = current;
        finish = (current + nchoosek(latticecol+1,2) - ...
            nchoosek(latticecol,2));
        rightedge = [start, finish]';
        listofhedges(1:2,size(listofhedges,2)+1) = rightedge;

    end
end

% Create vertical edges of the last column of the lattice
for current = (nchoosek(n,2)+1):nchoosek(n+1,2)-1

    % Edge going above
    start = current;
    finish = current+1;
    aboveedge = [start, finish]';
    listofvedges(1:2,size(listofvedges,2)+1) = aboveedge;

end

% Create a Hamiltonian of zeroes
vertexmax = nchoosek(n+1,2);
H = zeros(vertexmax);

% Update Hamiltonian entries for vertical edges
for p = 1:size(listofvedges,2);
    x = listofvedges(1,p);
    y = listofvedges(2,p);
```

```
    H(x,x) = H(x,x) + 1;
    H(y,y) = H(y,y) + lambda_v^2;
    H(x,y) = H(x,y) - lambda_v;
    H(y,x) = H(y,x) - lambda_v;
end

% Update Hamiltonian entries for horizontal edges
for p = 1:size(listofhedges,2);
    x = listofhedges(1,p);
    y = listofhedges(2,p);
    H(x,x) = H(x,x) + 1;
    H(y,y) = H(y,y) + lambda_h^2;
    H(x,y) = H(x,y) - lambda_h;
    H(y,x) = H(y,x) - lambda_h;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       Ham2.m
%%% Function:   Ham2(n, lambda_v, lambda_h)
%%% Purpose:    Creates the Hamiltonian operator for a lattice of n
%%% points and slope 2 with nearest-neighbor interaction weights
%%% lambda_v and lambda_h.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function H = Ham2(n, lambda_v, lambda_h)

% Initialize vector of edges
listofhedges = [1;2];
listofvedges = [];

% Loop through each column of the lattice except the last; this ...
    gives all edges besides vertical edges for outermost column of ...
    lattice
for latticecol = 2:(n-1)

    % Edge for boundary vertex
    boundary = latticecol^2;                                        ...
        % Calculate the boundary point

        % Edge going to the right
        start = boundary;                                           ...
            % Calculate the starting point
        finish = (boundary + (latticecol^2 - (latticecol-1)^2));    ...
            % Calculate the ending point
        boundaryedge = [start, finish]';                            ...
            % Put together the edge
        listofhedges(1:2,size(listofhedges,2)+1) = boundaryedge;    ...
            % Append the edge to the list of edges

    % Edges for non-boundary vertices
    for current = ((latticecol-1)^2+1):(latticecol^2-1)

        % Edge going above
        start = current;
        finish = current+1;
        aboveedge = [start, finish]';
```

```
        listofvedges(1:2,size(listofvedges,2)+1) = aboveedge;

        % Edge going to the right
        start = current;
        finish = (current + latticecol^2 - (latticecol-1)^2);
        rightedge = [start, finish]';
        listofhedges(1:2,size(listofhedges,2)+1) = rightedge;

    end
end

% Create vertical edges of the last column of the lattice
for current = ((n-1)^2+1):(n^2)-1

    % Edge going above
    start = current;
    finish = current+1;
    aboveedge = [start, finish]';
    listofvedges(1:2,size(listofvedges,2)+1) = aboveedge;

end

% Create a Hamiltonian of zeroes
vertexmax = n^2;
H = zeros(vertexmax);

% Update Hamiltonian entries for vertical edges
for p = 1:size(listofvedges,2)
    x = listofvedges(1,p);
    y = listofvedges(2,p);
    H(x,x) = H(x,x) + 1;
    H(y,y) = H(y,y) + lambda_v^2;
    H(x,y) = H(x,y) - lambda_v;
    H(y,x) = H(y,x) - lambda_v;
end

% Update Hamiltonian entries for horizontal edges
for p = 1:size(listofhedges,2)
    x = listofhedges(1,p);
    y = listofhedges(2,p);
    H(x,x) = H(x,x) + 1;
    H(y,y) = H(y,y) + lambda_h^2;
    H(x,y) = H(x,y) - lambda_h;
    H(y,x) = H(y,x) - lambda_h;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       GroundState.m
%%% Function:   GroundState(H)
%%% Purpose:    Returns the ground state vector (i.e., that which
%%% corresponds to the smallest eigenvalue by absolute value) of a
%%% given matrix H.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function groundstate = GroundState(H)
```

```
% Diagonalize H
[V, D] = eig(H);

% Sort the V and D matrices in ascending order
[discard, permutation]=sort(abs(diag(D)));
D = D(permutation, permutation);
V = V(:, permutation);

% Return the ground state eigenvector
groundstate = V(:,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       NormRatio.m
%%% Function:   NormRatio(vector)
%%% Purpose:    Computes, for a given vector, the norm ratio of
%%% (infinity-norm)/(2-norm).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = NormRatio(vector)

% Calculate infinity-norm divided by two-norm of the desired ...
    eigenvector
result = max(abs(vector)) / norm(vector,2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       ReturnSpectralGap.m
%%% Function:   ReturnSpectralGap(n, lambda_h_min, lambda_h_max,
%%%                 lambda_h_interval, lambda_v, linespec)
%%% Purpose:    Returns spectral gap values for a Hamiltonian with a
%%% slope 1 lattice of size n, for a given lambda_v and range of
%%% varying lambda_h values. Similar function for slope 2 Hamiltonians.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function evalues = ReturnSpectralGap(n, lambda_h_min, lambda_h_max, ...
    lambda_h_interval, lambda_v, linespec)

% Initialize eigenvalues array
evalues = [];

for lambda_h = lambda_h_min:lambda_h_interval:lambda_h_max

    % Compute the Hamiltonian and diagonalize
    H = Ham1(n, lambda_h, lambda_v);
    D = eig(H);

    % Take the absolute value of all eigenvalues
    D = abs(D);

    % Sort the eigenvalues in ascending order
    Dsorted = sort(D);

    % Select the second smallest eigenvalue by absolute value
    selected = D(2);

    % Add to the eigenvalues matrix
```

```
    evalues = [evalues D(2)];

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       PlotSpectralGap.m
%%% Function:   PlotSpectralGap(n, lambda_h_min, lambda_h_max,
%%%                 lambda_h_interval, lambda_v, linespec)
%%% Purpose:    Calculates and plots the spectral gap for the slope 1
%%% Hamiltonian of lattice size n, for a given lambda_v and varying
%%% lambda_h's and the desired linespec. Note: Should be used after the
%%% command "hold on". Similar function for slope 2 Hamiltonians.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotSpectralGap(n, lambda_h_min, lambda_h_max, ...
    lambda_h_interval, lambda_v, linespec)

% Initialize evalues and lambdas arrays
evalues = [];
lambdas = [];

for lambda_h = lambda_h_min:lambda_h_interval:lambda_h_max

    % Construct the Hamiltonian and diagonalize
    H = Ham1(n, lambda_h, lambda_v);
    D = eig(H);

    % Take the absolute value of all eigenvalues
    D = abs(D);

    % Sort the eigenvalues in ascending order
    Dsorted = sort(D);

    % Select the second smallest eigenvalue by absolute value
    selected = D(2);

    % Add to the evalues matrix
    evalues = [evalues D(2)];

end

% Plot the spectral gaps
plotname = sprintf('\\lambda_v = %g', lambda_v);
plot(lambda_h_min:lambda_h_interval:lambda_h_max, evalues, ...
    linespec,'DisplayName',plotname);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       PlotNormRatioByN.m
%%% Function:   PlotNormRatioByN(max_n, lambda_h, lambda_v, linespec)
%%% Purpose:    Calculates and plots the norm ratio of the ground state
%%% eigenvector for slope 1 Hamiltonians of sizes 2 to max_n, with
%%% given lambdas and the specified linespec. Note: Should be used
%%% after the command "hold on". Similar function for slope 2
%%% Hamiltonians
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function PlotNormRatioByN(max_n, lambda_h, lambda_v, linespec)

% Initialize results array
results = [];

% Perform computations
for n = 2:max_n

    % Calculate the Hamiltonian
    H = Ham1(n, lambda_h, lambda_v);

    % Compute the ground state
    groundstate = GroundState(H);

    % Calculate the ground state's (infinity-norm)/(2-norm)
    normratio = NormRatio(groundstate);

    % Add to results array
    results = [results normratio];

end

% Plot the results
plotname = sprintf('\\lambda_h = %g, \\lambda_v = %g', lambda_h, ...
    lambda_v);
plot(2:max_n, results, linespec, 'DisplayName', plotname);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       PlotNormRatioByLambdah.m
%%% Function:   PlotNormRatioByLambdah(n, lambda_h_min, lambda_h_max,
%%%                 lambda_h_interval, lambda_v, linespec)
%%% Purpose:    Computes and plots the norm ratio of the ground state
%%% for Hamiltonians with slope 1 lattices of sizes 2 to max_n,
%%% with the given lambdas. Note: Should be used after the command
%%% "hold on". Similar function for slope 2 lattices.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotNormRatioByLambdah(n, lambda_h_min, lambda_h_max, ...
    lambda_h_interval, lambda_v, linespec)

% Initialize results array
results = [];

% Perform computations
for lambda_h = lambda_h_min:lambda_h_interval:lambda_h_max

    % Calculate the Hamiltonian
    H = Ham1(n, lambda_h, lambda_v);

    % Compute the ground state
    groundstate = GroundState(H);

    % Calculate the ground state's (infinity-norm)/(2-norm)
    normratio = NormRatio(groundstate);
```

```
    % Add to results array
    results = [results normratio];

end

% Plot the results
plotname = sprintf('\\lambda_v = %g', lambda_v);
plot(lambda_h_min:lambda_h_interval:lambda_h_max, results, ...
    linespec,'DisplayName',plotname);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       DensityPlot1.m
%%% Function:   DensityPlot1(n, lambda_v, lambda_h, resolution, show,
%%%                 save)
%%% Purpose:    Creates a density plot indicating the probability of
%%% the particle being found at each site on the slope 1 lattice of
%%% size n, with Hamiltonian lambda parameters as specified. Plotting
%%% resolution was specified at 0.01 to generate plots in the paper.
%%% Probability is indicated by both size and color. If save == 1,
%%% saves the plot as a .jpg file. if show == 1, displays the image.
%%% Note: Please create the 'DensityPlot_results' folder in the Matlab
%%% current directory firstif saved plots are desired. Similar function
%%% for slope 2 lattices.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function h = DensityPlot1(n, lambda_v, lambda_h, resolution, show, ...
    save)

% Set up graphical environment
close all
h = 1;
figure(h);
set(h, 'color', 'white');
if show == 0
    set(h, 'Visible', 'off');
end
hold on

% Initialize results matrix
results = [];
plotnorm_list = [];

% Generate the list of vertices
lastvertex = nchoosek(n+1, 2);
L = VertexList1(lastvertex);

% Compute the Hamiltonian
H = Ham1(n, lambda_h, lambda_v);

% Diagonalize the Hamiltonian
[V, D] = eig(H);

% Generate rounded norms for plotting
for i = 1:length(V(1,:))

    % Calculate the norm squared
```

```
    normsq = norm(V(i), 2)^2;

    % Round the norm for plotting
    normsq_r = round(normsq * 1000) / 1000;
    plotnorm = 1000 * normsq_r;
    plotnorm_list = [plotnorm_list plotnorm];

end


% Set plotting parameters
size = linspace(2, 500, 1000);      % to vary circle size
color = linspace(1, 101, 1000);     % to vary circle color

% Plot each vertex
for i = 1:length(V(1,:))

    % Plot in a scatterplot
    scatter(L(2,i), L(3,i), size(plotnorm_list(i)+1), ...
        color(plotnorm_list(i)+1));
    axis([-1 n+1 -1 n+1])

end

% Plotting annotations
plotname = sprintf('Probability Density Plot for \\lambda_v = %g, ...
    \\lambda_h = %g (lattice columns = %g)', lambda_v, lambda_h, n);
% title(plotname);              % Uncomment if plot title is desired
xlabel('Lattice column number');
ylabel('Lattice row number');

% Save the figure
if save == 1
    filename = sprintf('densityplot_lv_%4.2f_lh_%4.2f_n_%g.jpg', ...
        lambda_v, lambda_h, n);
    fullname = fullfile(pwd,'DensityPlot_results',filename);
    I = getframe(h);
    set(h, 'units', 'inches', 'PaperPosition', [0 0 4 3]);
    imwrite(I.cdata, fullname, 'Quality',100);
end

% Reset graphical environment
hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       MultipleDensityPlots.m
%%% Function:   MultipleDensityPlots(n, lambda_h_min, lambda_h_max,
%%%                 interval_h, lambda_v_min, lambda_v_max, interval_v
%%%                 resolution)
%%% Purpose:    Creates density plots for a Hamiltonian with a slope
%%% 1 lattice of size n with a range of lambda_h and lambda_v values.
%%% Plotting resolution was specified at 0.01 to generate plots in the
%%% paper. Note: Please create the 'DensityPlot_results' folder in the
%%% Matlab current directory first. Similar function for slope 2
%%% lattices. Returns 1 if successful.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% Create density plots for a range of lambda_h and lambda_v values.
function success = MultipleDensityPlots(n, lambda_h_min, ...
    lambda_h_max, interval_h, lambda_v_min, lambda_v_max, ...
    interval_v, resolution)

% Reset graphical environment
close all;

% Create list of lambdas to use
lambda_h_list = lambda_h_min:interval_h:lambda_h_max;
lambda_v_list = lambda_v_min:interval_v:lambda_v_max;

% Replace all zeroes by small non-zero values
for i = 1:length(lambda_h_list)
    if lambda_h_list(i) == 0
        lambda_h_list(i) = 0.000001;
    end
    if lambda_v_list(i) == 0
        lambda_v_list(i) = 0.000001;
    end
end

i = 1;

% Compute the number of plots to be created
fprintf('A total of %g plots will be created and saved.\n', ...
    length(lambda_h_list)*length(lambda_v_list));

% Main execution loop; create a density plot for every pair of ...
    lambda_h, lambda_v
for lambda_h = lambda_h_list
    for lambda_v = lambda_v_list
        temp = DensityPlot1(n, lambda_v, lambda_h, resolution, 0, 1);
        fprintf('Plot number %g complete.\n', i);
        i = i+1;
    end
end

% Reset graphical environment
close all;

% Return success code
success = 1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       CreatePhaseDiagram.m
%%% Function:   CreatePhaseDiagram(n, lambda_h_min, lambda_h_max,
%%%                 interval_h, lambda_v_min, lambda_v_max, interval_v)
%%% Purpose:    Assists the user in creating phase diagrams for density
%%% plots created with MultipleDensityPlots. Saves the classification
%%% data in the file "classification.mat". Note: Arguments used should
%%% be the same as those used in the MultipleDensityPlots function. To
%%% recreate a phase diagram from the "classification.mat" file, use
%%% the function RecreatePhaseDiagram.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function results = CreatePhaseDiagram(n, lambda_h_min, ...
    lambda_h_max, interval_h, lambda_v_min, lambda_v_max, interval_v)

% Set up graphical environment
close all;

% Initialize results matrix
results = zeros(length(lambda_h_min:interval_h:lambda_h_max), ...
    length(lambda_v_min:interval_v:lambda_v_max));

% Note: lambda_h values vary across rows
%       lambda_v values vary across columns

% Main execution loop; display a density plot for every pair of ...
    lambda_h, lambda_v and request a classification

% First lambda_h goes into first row
results_row = 1;

for lambda_h = lambda_h_min:interval_h:lambda_h_max

    % First lambda_v goes into first column
    results_col = 1;

    for lambda_v = lambda_v_min:interval_v:lambda_v_max

        % Display density plot
        currentfile = ...
            sprintf('densityplot_lv_%4.2f_lh_%4.2f_n_%g.jpg', ...
            lambda_h, lambda_v, n);
        fullname = fullfile(pwd,'DensityPlot_results',currentfile);
        I = imread(fullname);
        imshow(I);

        % Request classification
        class = input('Please classify this density plot ...
            configuration:\n', 's');

        % Store classification
        results(results_row, results_col) = class;

        % Increment results column index
        results_col = results_col+1;

    end

    % Increment results row index
    results_row = results_row+1;

end

% Convert results matrix to a character array
results = char(results)

% Plot the phase diagram
close all;
```

```
figure;
hold on;
size(results)
j = 1;
for j = 1:size(results(:,1))            % Iterate through values of ...
    lambda_h
    k = 1;
    for k = 1:size(results(:,2))        % Iterate through values of ...
        lambda_v
        marker = 'o';
        if results(j,k) == 'c'          % Corner (near)
            color = 'm';
        elseif results(j,k) == 'd'      % Diagonal boundary
            color = 'y';
        elseif results(j,k) == 'f'      % Far corner
            color = 'c';
        elseif results(j,k) == 'r'      % Right boundary
            color = 'r';
        elseif results(j,k) == 'b'      % Bottom boundary
            color = 'g';
        elseif results(j,k) == 's'      % Scattered
            color = 'k';
        elseif results(j,k) == 'e';     % Bottom right corner
            color = 'b';
        else
            color = 'k';
            marker = 'd';
        end
        scatter(j, k, color, marker);
        k = k+1;
    end
    j = j+1;
end

% Save data
save('classification.mat', 'results');

% Reset graphical environment
hold off;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% File:       RecreatePhaseDiagram.m
%%% Function:   RecreatePhaseDiagram(n, lambda_h_min, lambda_h_max,
%%%                 interval_h, lambda_v_min, lambda_v_max,
%%%                 interval_v, results)
%%% Purpose:    Recreates a phase diagram from density plots created
%%% previously with MultipleDensityPlots and classified with
%%% CreatePhaseDiagram. Loads the user classification data from the
%%% array results. Note: Load the 'classification.mat' data first and
%%% specify the loaded variable name as parameter "results". Other
%%% arguments used should be the same as those originally used in the
%%% MultipleDensityPlots function.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function results = RecreatePhaseDiagram(n, lambda_h_min, ...
    lambda_h_max, interval_h, lambda_v_min, lambda_v_max, ...
```

```
    interval_v, results)

% Set up graphical environment
close all;
h = 1;
figure(h);
hold on;

% Plot the phase diagram
j = 1;                                  % Start at row 1
results = transpose(results);           % Correct which one is ...
    lambda_h, which one is lambda_v
for j = 1:size(results(:,1))            % Iterate through values of ...
    lambda_h
    k = 1;
    for k = 1:size(results(:,2))        % Iterate through values of ...
        lambda_v
        marker = 'o';
        if results(j,k) == 'c'          % Corner (near)
            color = 'g';
            marker = 'o';
        elseif results(j,k) == 'd'      % Diagonal boundary
            color = 'y';
            marker = 'x';
        elseif results(j,k) == 'f'      % Far corner
            color = 'b';
            marker = 'square';
        elseif results(j,k) == 'r'      % Right boundary
            color = 'm';
            marker = 'pentagram';
        elseif results(j,k) == 'b'      % Bottom boundary
            color = 'c';
            marker = '*';
        elseif results(j,k) == 's'      % Scattered
            color = 'k';
            marker = 'd';
        elseif results(j,k) == 'e';     % Bottom right corner
            color = 'r';
            marker = '^';
        else                            % Errors
            color = 'k';
            marker = 'd';
        end
        scatter(j, k, color, marker);
        k = k+1;
    end
    j = j+1;
end

% Set axes and background color
set(findobj(gcf,'type','axes'), 'XTickLabel', [0:0.2:lambda_h_max + ...
    0.1])
set(findobj(gcf,'type','axes'), 'YTickLabel', [0:0.2:lambda_v_max + ...
    0.1])
set(gcf,'color','w');

% Annotate plot
```

```
xlabel('\lambda_h');
ylabel('\lambda_v');
title('Phase Diagram of Density Plot State by \lambda_h, \lambda_v');

% Reset graphics environment
hold off;
```