Spring 2021

Electronic Properties of Flat and Curved Graphene Sheets

Deng Yanpei
Bard College

Recommended Citation
Yanpei, Deng, "Electronic Properties of Flat and Curved Graphene Sheets" (2021). *Senior Projects Spring 2021*. 203.
https://digitalcommons.bard.edu/senproj_s2021/203

Electronic Properties of Flat and Curved Graphene Sheets

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Yanpei Deng

Annandale-on-Hudson, New York
May, 2021

Abstract

This paper explored the electronic properties of the graphene sheet and also developed basis for understanding the electronic properties of the curved graphene sheet. This paper began with setting up basic knowledge about solid-state physics including introducing band structure, band gap, crystal structure, and reviews for quantum mechanical operators. Then this paper described two potential models that are suitable for considering periodic potential: the weak potential and the tight-binding model. We discovered the tight-binding model is better for our graphene case and by applying this model we find the energies of the graphene sheet. Next, we constructed the 1D and 2D Hamiltonian matrix in python with periodic potentials and plotted the energy levels and wave functions. Finally, this paper touched on gaussian curvatures and two possible shapes for graphene to be curved.

Contents

Dedication

To my loving parents and my brother Jinchun Deng, who has a great potential to become a future physicist.

Acknowledgments

I would like to thank my advisors Professor Harold Haggard and Professor Paul Cadden-Zimansky for their patient guidances that helped me finish this project. I would give special thanks to Professor Matthew Deady for his kindness support and personal notes he shared to instruct me on reading hard references. Professor Matthew Deady also guided me to major in physics, so thanks him for leading me to this attractive field and let me find my life goal. Thanks Professor Shuo Zhang and Professor Ethan Bloch for their advice on how to organize paper in a professional way.

Thanks to my family members, who are distant away but always support me mentally and encourage me to keep working on this project.

Thanks to my friends: Jiangli Liu, Lixue Zeng, Vera Lu, Beida Liang, Zihan(Edward) Li, Nathalie Jones, Andrew Poverman, Xiyuan Chen and Weiting Sun for the warmth they gave me during this hard time.

1

Introduction

In 2005 Andre Geim's group and Philip Kim and Horst Stormer's group have discovered the quantum Hall effect of graphene. Since then graphene has become a popular topic to study in condensed-matter physics. Graphene has a great potential for applications in industry such as making phone chips. It is possible to control the carrier density in graphene sheet by simple application of a gate voltage, and this effect is a fundamental element for the design of electrical devices.[5] Hence I am interested in studying graphene. There are also some personal reasons: I have been working in the Bard Nanolab under the instruction of Professor Paul Cadden-Zimansky on graphene fabrication and laser ablation. I joined the Bard Summer Research Institute in 2019 and with this group, I attended two American Physics Society's meetings. These experiences all strengthened my interests in knowing the mystery of graphene. I found myself is familiar with the experiments but I never understand the theory behind it, so I have a strong desire to dedicate this project to study the electronic properties of graphene. This project is for juniors who also interested in condensed-matter physics and I hope it will help them set up basic understanding of solid state physics.

In this paper, I began with introducing the basic knowledge of solid state physics: crystal structure and the band structure. Under these two big sections, I include descriptions about Bravais lattice, reciprocal lattice, Brillouin zone and how they look like in graphene; then I introduce the band structures and band gap with its application in identifying the resistance. I also proved one theorem that we used throughout the project, Bloch's theorem. Then I discussed two models: the weak potential and the tight-binding model. Then by comparison, we found the tight-binding model is more suitable for graphene and by applying this model, we find the energy levels of graphene sheet. Next we constructed 1D and 2D Hamiltonian matrix with periodic potentials in Python with plottings of energy levels and energy eigenstates. I developed Python codes over the last two semesters and have shared them in the appendix. Finally, this paper introduced Gaussian curvature and discussed future thoughts on how to construct the curved graphene sheets' Hamiltonian.

2

Basic Knowledge of Solid

2.1 Crystal Structure

In this section, we are going to discuss the crystal structure: a group of atoms attached to every lattice point, is called the basis; when repeated in space it forms the crystal structure. [4]

2.1.1 Reciprocal Lattice

There are three translational vectors $\vec{a_1}, \vec{a_2}, \vec{a_3}$ and lattice defined by these vectors. When viewed from a random point $\vec{r}$, defined by

$$\vec{r'} = \vec{r} + u_1\vec{a_1} + u_2\vec{a_2} + u_3\vec{a_3} \tag{2.1.1}$$

where u_1, u_2, u_3 are integers, it should look the same as if you see from other directions. Hence, the set of $\vec{r'}$ defined the lattice. If $\vec{r}$ and $\vec{r'}$ have the same atomic arrangement, lattice and $\vec{a_1}, \vec{a_2}, \vec{a_3}$ are primitive. Then we can introduce few definitions before we fully explained the reciprocal lattce.

"The parallelepiped defined by primitive axes, $\vec{a_1}, \vec{a_2}, \vec{a_3}$ is called a **primitive cell**. One specific example, is the **Wigner-Seitz** cell: it can choose to be full symmetry about a lattice point is the region of space that is closer to that point then to any other lattice point. And a **Bravais Lattice** is an infinite array of discrete points with an arrangement and orientation that appears exactly the same, from whichever of the points the array is viewed." [4]

Let a set of $\vec{R}$(points) constituting a Bravais lattice and a set of all wave vector $\vec{K}$ yield plane waves with periodicity of a given Bravais lattice is known as its reciprocal lattice. With

$$e^{i\vec{K}\cdot(\vec{r}+\vec{R})} = e^{i\vec{K}\cdot\vec{r}} \tag{2.1.2}$$

$$e^{i\vec{K}\cdot\vec{R}} = 1. \tag{2.1.3}$$

Again let $\vec{a_1}, \vec{a_2}, \vec{a_3}$ be a set of primitive vectors. The reciprocal lattice can be presented as:

$$\vec{b_1} = 2\pi\frac{\vec{a_2}\times\vec{a_3}}{\vec{a_1}\cdot(\vec{a_2}\times\vec{a_3})} \tag{2.1.4}$$

$$\vec{b_2} = 2\pi\frac{\vec{a_3}\times\vec{a_1}}{\vec{a_1}\cdot(\vec{a_2}\times\vec{a_3})} \tag{2.1.5}$$

$$\vec{b_3} = 2\pi\frac{\vec{a_1}\times\vec{a_2}}{\vec{a_1}\cdot(\vec{a_2}\times\vec{a_3})}. \tag{2.1.6}$$

And the relation between $\vec{a}$s and $\vec{b}$s are $\vec{b_i}\cdot\vec{a_j} = 2\pi\delta_{ij}$. So when $i = j$, $\delta_{ij} = 1$ and when $i \neq j$, $\delta_{ij} = 0$.

When we compute $\vec{b_1}\cdot(\vec{b_2}\times\vec{b_3})$, we get $\frac{(2\pi)^3}{\vec{a_1}\cdot(\vec{a_2}\times\vec{a_3})}$, by using the facts that $\vec{A}\cdot(\vec{B}\times\vec{C}) = \vec{B}\cdot(\vec{C}\times\vec{A})$ and $\vec{A}\times(\vec{B}\times\vec{C}) = \vec{B}(\vec{A}\cdot\vec{C} - \vec{C}(\vec{A}\cdot\vec{B})$.

2.1.2 Operators

We take a side way here to discuss how to determine wether a wave function is a momentum eigenstate or not. In a traditional way, we usually apply momentum operator to the wave function:

$$\hat{P} = -i\hbar\frac{\partial}{\partial x}.$$

We have two wave functions:

$$\psi = A\sin(wt + kx), \tag{2.1.7}$$

and

$$\psi = e^{i(kx+wt)}. \tag{2.1.8}$$

We act $\hat{P}$ on 2.1.7 and we get:

$$\hat{P}\psi = Ak\cos(wt + kx).$$

We did not get a constant times the same wave function, so the wave function is not a momentum eigenstate. We can get this result also by understanding whether it has translational invariance or not. By times the conjugate to the original wave function, we get the probability of where to find the particle. For 2.1.7, we times it with its conjugate and get: $\psi = A^2 \sin^2(wt + kx)$. We plot the sine square graph. If we shifted shifted the graph, the probability will not remain the same. Therefore, this wave function does not have translational invariance, therefore, it will not be a momentum eigenstate.

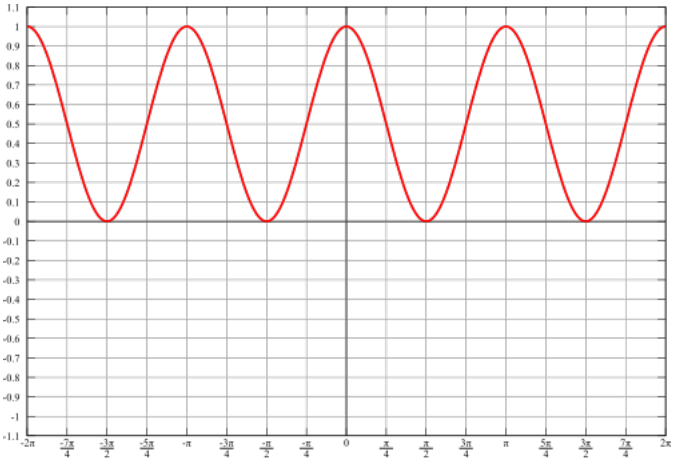


Figure 2.1.1. If we shifted the sine square curve, the probability of finding the particle will change too.

We can try also for 2.1.8 Apply the $\hat{P}$:

$$\hat{P}\psi = -i\hbar e^{i(kx+wt)} ik = \hbar k e^{i(kx+wt)}.$$

We get a constant times the original wave function, therefore, the wave function is a momentum eigenstate. For 2.1.8, we times it with its conjugate and get: $\psi = e^{i(kx+wt)} e^{-i(kx+wt)} = 1$. We plot the $y = 1$ graph.

If we shifted shifted the graph, the probability will remain the same. Therefore, this wave function has translational invariance, therefore, it is a momentum eigenstate.

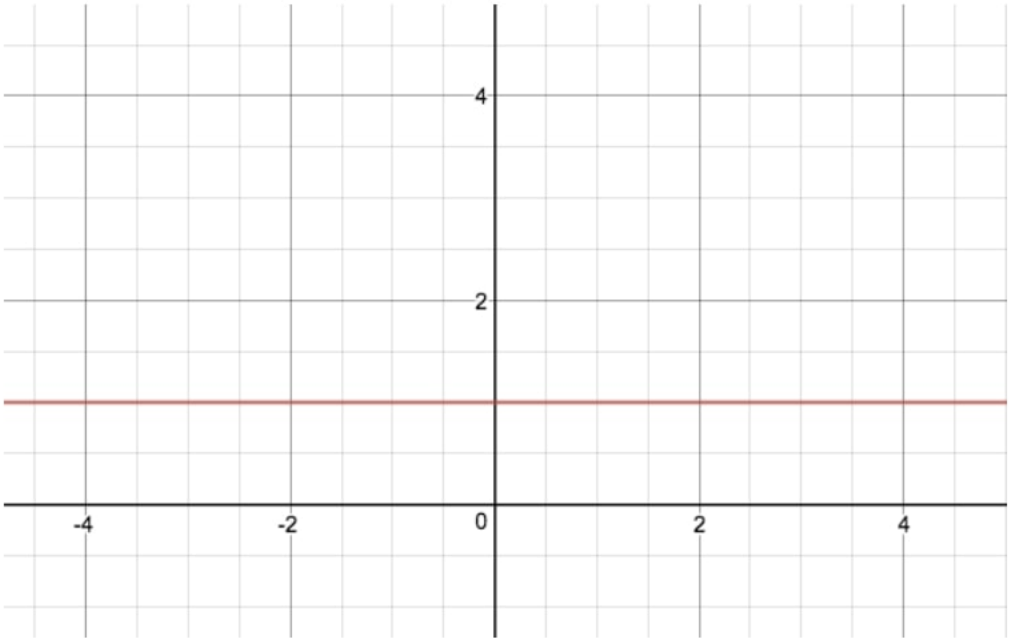


Figure 2.1.2. This is a momentum eigenstate.

2.1.3 Crystal Structure of Graphene

In this section, let us zoom in to the crystal structure of graphene. The carbon atoms within graphene form the honeycomb structure. From Fig 2.1.3(a) shows the honeycomb lattice of graphene. And A and B in Fig 2.1.3(a) are two sublattices. The distance between two neighbor carbon atoms is 0.142nm. In Fig 2.1.3(b), we can review the triangular Braivais lattice with two-atom basis. The three vectors $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ and $\boldsymbol{\delta}_3$ are connecting sublattice A to sublattice B and they are defining by:

$$\boldsymbol{\delta}_1 = \frac{a}{2}(\sqrt{3}\mathbf{e}_x + \mathbf{e}_y), \tag{2.1.9}$$

$$\boldsymbol{\delta}_2 = \frac{a}{2}(-\sqrt{3}\mathbf{e}_x + \mathbf{e}_y), \tag{2.1.10}$$

$$\boldsymbol{\delta}_3 = -a\mathbf{e}_y, \tag{2.1.11}$$

where $\mathbf{e}_x$ and $\mathbf{e}_y$ are basis vectors. $\mathbf{a}_1$ and $\mathbf{a}_2$ are defined by:

$$\mathbf{a}_1 = \sqrt{3}a\mathbf{e}_x \text{ and } \mathbf{a}_2 = \frac{\sqrt{3}a}{2}(\mathbf{e}_x + \sqrt{3}\mathbf{e}_y). \tag{2.1.12}$$

Finally, "the reciprocal lattice, which is defined with respect to the triangular Bravais lattice, is depicted in Fig 2.1.3(b)."[5] It is spanned by the vectors:

$$\mathbf{a}_1^* = \frac{2\pi}{\sqrt{3}a}\left(\mathbf{e}_x - \frac{\mathbf{e}_y}{\sqrt{3}}\right) \text{ and } \mathbf{a}_2^* = \frac{4\pi}{3a}\mathbf{e}_y. \tag{2.1.13}$$

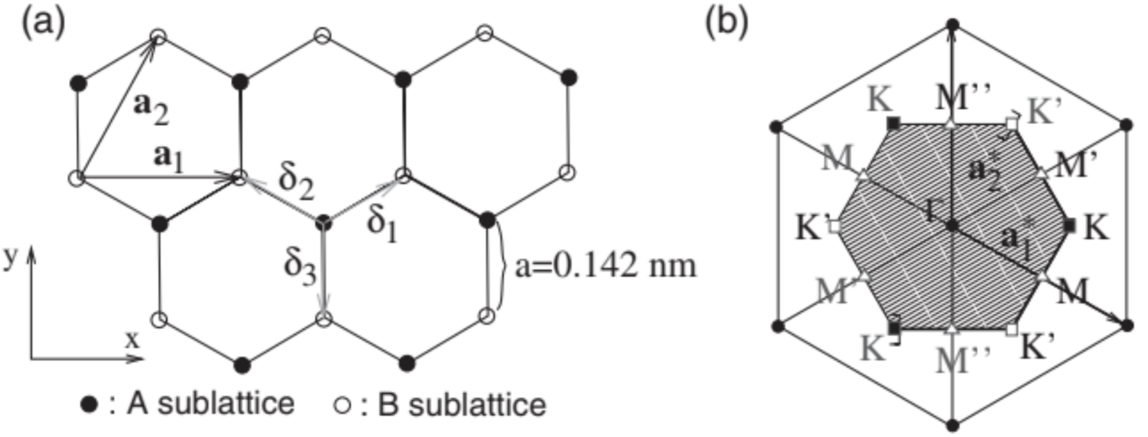


Figure 2.1.3. is from [5]. (a)Shows the honeycomb lattice and (b) shows the triangular Bravais lattice.

2.1.4 Boundary Conditions

I mentioned in the previous proof of Bloch's theorem that the plane wave satisfies the Born-von Karman boundary conditions and now we are focusing on how what is this boundary condition. The Born-von Karman boundary condition is about macroscopic periodicity, which is defined by:

$$\psi(x, y, z + L) = \psi(x, y, z)$$

$$\psi(x, y + L, z) = \psi(x, y, z)$$

$$\psi(x + L, y, z) = \psi(x, y, z),$$

with the Bravais lattice is cubic and L is an integral multiple of the lattice constant a. However, not all the Bravais lattices are cubic and for general application, we consider working with a primitive cell of the underlying Bravais lattice. That is if we want to move one lattice, we move a, if we want to move over one crystal. we move $N_i\mathbf{a}_i$ for $\mathbf{a}$ can be in any directions and $i = 1, 2, 3,$. Then we can rewrite the boundary condition to be:

$$\psi(\mathbf{r} + N_i\mathbf{a}_i) = \psi(\mathbf{r}). \tag{2.1.14}$$

Then we apply the Bloch's theorem $\psi(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}\psi(\mathbf{r})$ to 2.1.14:

$$\psi(\mathbf{r} + N_i\mathbf{a}_i) = e^{iN_i\mathbf{k}\cdot\mathbf{a}_i}\psi(\mathbf{r}).$$

That is to say, the exponential term must satisfy the following condition:

$$e^{iN_i\mathbf{k}\cdot\mathbf{a}_i} = 1,$$

with $i = 1, 2, 3,$. We also defines $\mathbf{k}$ to be $\mathbf{k} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3$. And then by substituting $\mathbf{k}$ in, we get:

$$e^{2\pi iN_i\mathbf{k}\cdot\mathbf{x}_i} = 1.$$

Therefore we can conclude that:

$$x_i = \frac{m_i}{N_i},$$

for m_i integral. Thus the general form for allowed Bloch wave vectors is:

$$\mathbf{k} = \sum_{i=1}^{3} \frac{m_i}{N_i}\mathbf{b}_i. \tag{2.1.15}$$

2.2 Band Structure

We want to study the electronic properties of graphene, and one important factor to consider would be the band structure. The Electronic Band Structure of a solid describes the range of energy for each allowed state that electrons may have within it. These states also have particular periodicity. The band structure can be thought of as an energy-momentum map. To understand band structure of a solid, we need to first understand the one-dimensional Dirac Comb and describe the Dirac Comb limit of the "Kronig-Penny Model". One specific quality that solids can be described using periodic potentials that repeat after some fixed distance a:

$$V(x + a) = V(x). \tag{2.2.1}$$

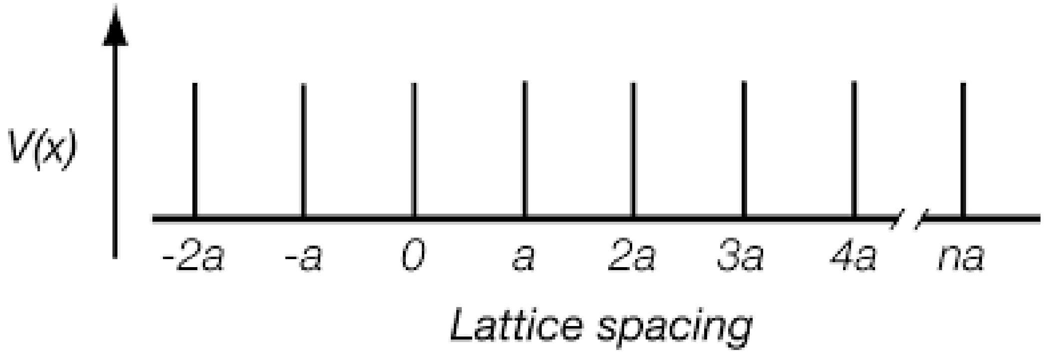


Figure 2.2.1. is from [2]. The potential tells us that there is a long string of delta-function spikes. These spikes are the one-dimensional Kronig-Penny model.

By Bloch's Theorem, the solution to the *Schödinger* equation has the property that:

$$\psi(x+a) = e^{iqa}\psi(x). \tag{2.2.2}$$

However, no realistic solid truly goes on forever, so at the end of a solid structure, the boundary condition has changed. One trick is to wrap the x axis around in a circle and connect it onto its tail, so this structure has no end and we can apply a periodic boundary condition. Then, the wave function repeats N times. The Dirac Comb potential is:

$$V(x) = \alpha \sum_{j=0}^{N-1} \delta(x - ja). \tag{2.2.3}$$

From the figure, we can see that when $0 < x < a$, the potential is zero. Therefore,

$$\frac{d^2\psi}{dx^2} = -k^2\psi, \text{with } k \equiv \frac{\sqrt{2mE}}{\hbar}. \tag{2.2.4}$$

Then the general solution is just:

$$\psi(x) = A\sin(kx) + B\cos(kx) \qquad (0 < x < a). \tag{2.2.5}$$

According to Bloch's Theorem, the wave function from $-a < x < 0$ is:

$$\psi(x) = e^{-iqa}\{A\sin[k(x+a)] + B\cos[k(x+a)]\}. \tag{2.2.6}$$

Since $\psi(x) = e^{-iqa}\{A\sin[k(x+a)] + B\cos[k(x+a)]\}$ is to the left of the initial region, when the original wave equation is at $x = 0$, here x is at $-a$ and ψ must be continuous, therefore,

$$B = e^{-iqa}[Asin(ka) + Bcos(ka)]. \tag{2.2.7}$$

Moreover, because the derivative of ψ suffers a discontinuity proportional to the delta function($\Delta(\frac{d\psi}{dx}) = -\frac{2m\alpha}{\hbar^2}\psi(0)$), we get

$$kA - e^{-iqa}k[A\cos(ka) - B\sin(ka)] = \frac{2m\alpha}{\hbar^2}B. \tag{2.2.8}$$

There is a sign change, because we have spikes instead of wells. By solving 2.2.7, we get expressions for $A\sin(ka)$ and by plugging that back into 2.2.8, we can cancel kB and hence simplify to:

$$\cos(qa) = \cos(ka) + \frac{m\alpha}{\hbar^2 k}\sin(ka), \tag{2.2.9}$$

for $q = \frac{2\pi n}{Na}, (n = 0, \pm1, \pm2, ...)$. By defining $z \equiv ka$ and $\beta \equiv \frac{m\alpha a}{\hbar^2}$, we can rewrite 2.2.9:

$$f(z) \equiv \cos(z) + \beta\frac{\sin(z)}{z}. \tag{2.2.10}$$

By plotting $f(z)$ vs. z, we get that there are only certain energies that are allowed the particles to be in. Some closely packed allowed energies form a band and together we get the band structure with gaps existing between each band.

2.2.1 The Use of Band Structure

After gaining the basic knowledge of band structure, we want to further understand what things the band structure can tell us. The band structure can help us define the conductivity of a material. For a conductor, the highest band is partially filled; whereas for an insulator, the highest band is completely filled. And before introducing the semiconductor, I would like to introduce the valence band and the conduction band first. The **valence band** is completely filled at zero temperature, and the band just above it is the **conduction band**. If a material is an insulator at zero temperature, but the gap between the valence band and the conduction band is small (smaller than 2eV), it is known as a **semiconductor**.[3] By understanding the band structure of a material, we can tell its electric conductivity.

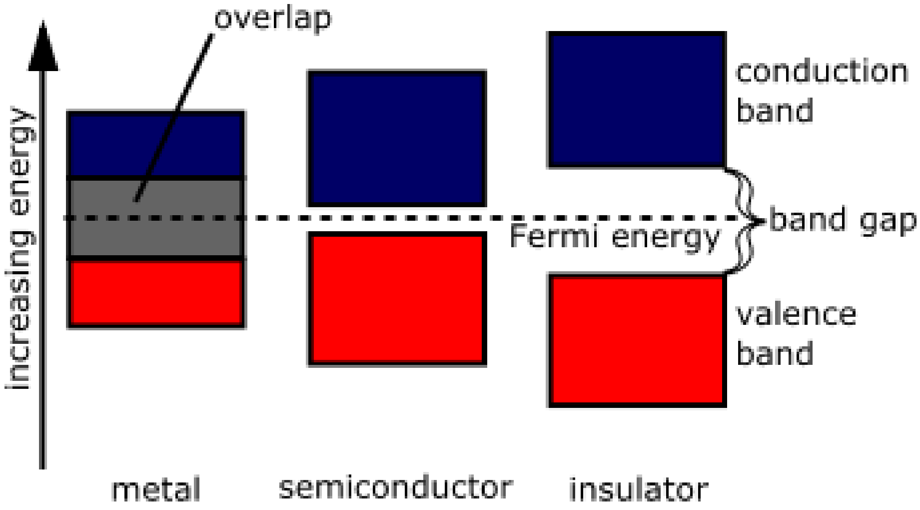


Figure 2.2.2. is from [13]. For conductor, there is an overlap between conduction band and valence band; for semiconductor the band between valence band and conduction band is small; for insulator, the gap is big.

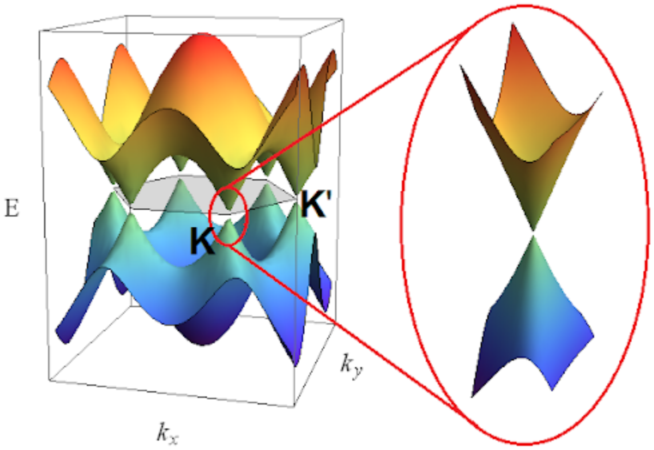


Figure 2.2.3. is from [10].This is the band structure of graphene. The touch points are where valence band and conduction band meet and form Dirac cones.

From this figure, we can see the orange region is the conduction band and the blue region is the valence band. And when the two bands meet, it forms a hexagonal Brillouin zone. Then let's discuss what is the Brillouin zone. We usually consider the 1st Brillouin zone to be the most important factor to understand. Brillouin zone is the primitive cell of the reciprocal lattice. We can confine all the periodicities of the lattice within the first Brillouin zone. So by understanding the electronic properties within the first Brillouin zone, we can understand the electronic properties of the whole lattice.

2.2.2 The Proof of Bloch's Theorem

We've been using Bloch's Theorem for a long time and now we want to understand how we prove Bloch's Theorem. This proof is following Ashcroft and Mermin.[1] We first consider a Fourier expansion of periodic potential and an associated wave function:

$$U(\mathbf{r}) = \sum_{\mathbf{K}} U_{\mathbf{K}}\, e^{i\mathbf{K}\cdot\mathbf{r}}, \tag{2.2.11}$$

$$\psi(\mathbf{r}) = \sum_{\mathbf{q}} c_{\mathbf{q}}\, e^{i\mathbf{q}\cdot\mathbf{r}}. \tag{2.2.12}$$

Then we want to find the Fourier coefficient of $U(\mathbf{r})$, that is:

$$U_{\mathbf{K}} = \frac{1}{v}\int_{\text{cell}} d\mathbf{r}\, e^{-i\mathbf{K}\cdot\mathbf{r}}\, U(\mathbf{r}).$$

Contrast standard Fourier expansion, there are two changes: first is to require the potential energy all over the primitive cell is zero ($U_0 = \frac{1}{v}\int_{\text{cell}} d\mathbf{r}\, U(\mathbf{r}) = 0$) and second we choose an origin at a symmetry point, so that $U(-\mathbf{r}) = U(\mathbf{r})$ (inversion symmetry). Since $U(\mathbf{r})$ is real, then $U_{\mathbf{K}}$ is also real, which means $U_{-\mathbf{K}} = U_{\mathbf{K}} = U^*_{\mathbf{K}}$. Then we consider the Schödinger's equation:

$$-\frac{\hbar^2}{2m}\nabla^2\, \psi(\mathbf{r}) + U(\mathbf{r})\, \psi(\mathbf{r}) = \mathcal{E}\, \psi(\mathbf{r}).$$

By algebraic manipulations and substituting the Fourier expansions of $\psi(\mathbf{r})$ and $U(\mathbf{r})$, we get the result:

$$\begin{aligned} -\frac{\hbar^2}{2m}\nabla^2\, \psi &= \sum_{\mathbf{q}} \frac{\hbar^2}{2m} q^2\, c_{\mathbf{q}}\, e^{i\mathbf{q}\cdot\mathbf{r}} \quad \text{and} \\ U\,\psi &= \left(\sum_{\mathbf{K}} U_{\mathbf{K}}\, e^{i\mathbf{K}\cdot\mathbf{r}}\right)\left(\sum_{\mathbf{q}} c_{\mathbf{q}}\, e^{i\mathbf{q}\cdot\mathbf{r}}\right) \\ &= \sum_{\mathbf{K}}\sum_{\mathbf{q}} [U_{\mathbf{K}}\, c_{\mathbf{q}}\, e^{-i(\mathbf{K}+\mathbf{q})\cdot\mathbf{r}}]. \end{aligned}$$

Defining $\mathbf{q}' = \mathbf{K} + \mathbf{q}$, we get $U\,\psi = \sum_{\mathbf{K}\mathbf{q}'} U_{\mathbf{K}}\, c_{\mathbf{q}'-\mathbf{K}} e^{i\mathbf{q}'\cdot\mathbf{r}}$. Then we simply change the indexing from $\mathbf{q}'$ and $\mathbf{K}$ to $\mathbf{q}$ and $\mathbf{K}'$ and plug into the Schödinger's equation:

$$\sum_{\mathbf{q}} e^{i\mathbf{q}\cdot\mathbf{r}}\left\{\left(\frac{\hbar^2}{2m}\, q^2 - \mathcal{E}\right) c_{\mathbf{q}} + \sum_{\mathbf{K}'} U_{\mathbf{K}'}\, c_{\mathbf{q}-\mathbf{K}'}\right\} = 0.$$

Since plane waves satisfying the Born-von Karman boundary condition are orthogonal, the separation coefficients($c_{\mathbf{q}}$, $U_{\mathbf{K}'}$) will vanish, therefore, we have:

$$\left(\frac{\hbar^2}{2m}\,q^2-\mathcal{E}\right)c_{\mathbf{q}}+\sum_{\mathbf{K}'}U_{\mathbf{K}'}\,c_{\mathbf{q}-\mathbf{K}'}=0.$$

Here we do another rewriting of the indices. Because $\mathbf{q}$ is any wave vectors, we define $\mathbf{q}=\mathbf{k}-\mathbf{K}$, where $\mathbf{k}$ lies in the first Brillouin zone and $\mathbf{K}$ represents the reciprocal lattice. Because the lattice is repeating, we can get to anywhere in the lattice by defining $\mathbf{q}=\mathbf{k}-\mathbf{K}$. Therefore, we can constrain the problem to the first Brillouin zone and our result apply anywhere else in the lattice. Thus, the Schödinger's equation becomes:

$$\left(\frac{\hbar^2}{2m}\,(\mathbf{k}-\mathbf{K})^2-\mathcal{E}\right)c_{\mathbf{k}-\mathbf{K}}+\sum_{\mathbf{K}'}U_{\mathbf{K}'}\,c_{\mathbf{q}-\mathbf{K}'}=0.$$

Here comes the third change in indexing, that is to write $\mathbf{K}'\to\mathbf{K}'-\mathbf{K}$. Thus:

$$\left(\frac{\hbar^2}{2m}\,(\mathbf{k}-\mathbf{K})^2-\mathcal{E}\right)c_{\mathbf{k}-\mathbf{K}}+\sum_{\mathbf{K}'}U_{\mathbf{K}'-\mathbf{K}}\,c_{\mathbf{q}-\mathbf{K}'}=0.$$

Therefore, we get the wave function as:

$$\psi_{\mathbf{k}}=\sum_{\mathbf{K}}c_{\mathbf{k}-\mathbf{K}}\,e^{i(\mathbf{k}-\mathbf{K})\cdot\mathbf{r}}. \tag{2.2.13}$$

Factoring the $e^{i\mathbf{k}\cdot\mathbf{r}}$ out:

$$\psi_{\mathbf{k}}(\mathbf{r})=e^{i\mathbf{k}\cdot\mathbf{r}}\left(\sum_{\mathbf{K}}c_{\mathbf{k}-\mathbf{K}}\,e^{-i\mathbf{K}\cdot\mathbf{r}}\right).$$

This is the Bloch's Theorem with a periodic function $u(\mathbf{r})$:

$$u(\mathbf{r})=\sum_{\mathbf{K}}c_{\mathbf{k}-\mathbf{K}}\,e^{-i\mathbf{K}\cdot\mathbf{r}}. \tag{2.2.14}$$

This completes the proof of Bloch's Theorem.

2.3 Band Gap

Here I want to further discuss the electronic band and gap. This discussion follows Professor Matthew Deady's personal notes. First let us consider "single atom electron states". That is, we will consider a single atom in a finite square well and its attracting electrons. The well has width

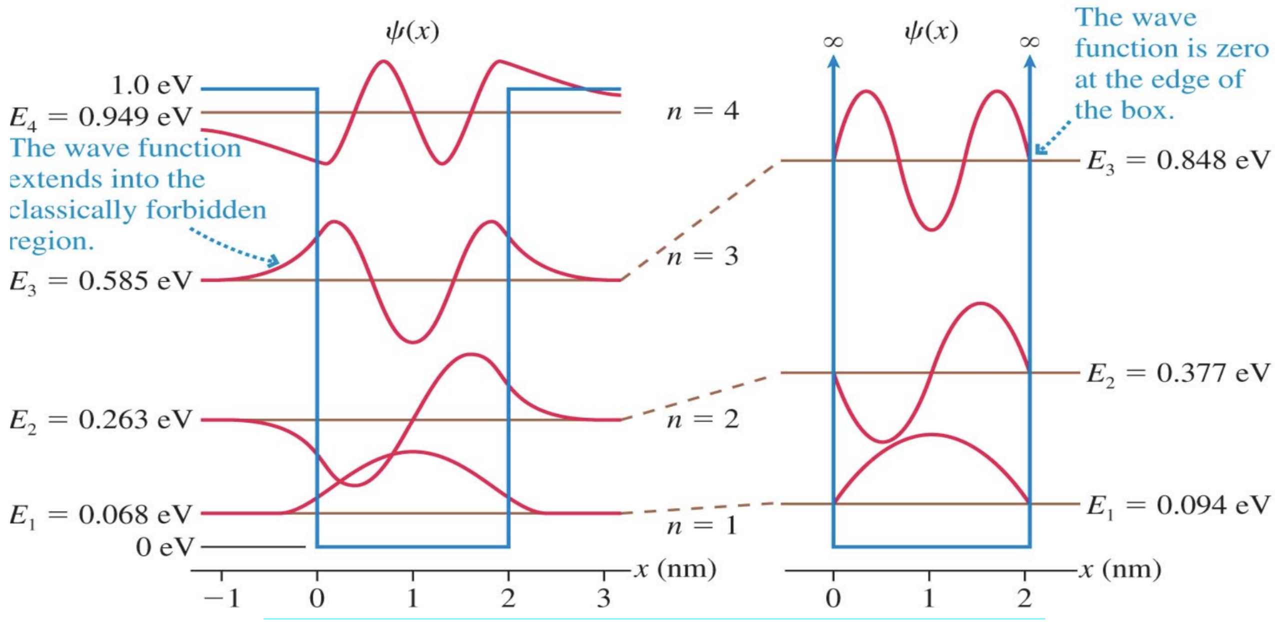


Figure 2.3.1. is from [9]. On the left shows the wave functions out side the finite square well; on the right shows the wave functions within the finite square well.

L and depth $-V_0$. The electron states are presented below. Inside the well, the time-independent Schrödinger's Equation needs:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} - V_0\psi(x) = E\psi(x),$$

$$\frac{d^2\psi}{dx^2} = -\left(\frac{\hbar^2}{2m}\right)(E + V_0)\psi(x).$$

For negative "bound state" energies $-V_0 < E < 0$, this gives $\psi'' = -\psi$ meaning that the curvature of the wave function is opposite the magnitude and the wave function looks vaguely sinusoidal inside the wells. When outside of the wells, there is no potential, but E is still negative, so $\psi'' = +\psi$, meaning the curvature has the same sign as the wave function. This indicates that when outside the well, the wave function looks vaguely exponential, sloping toward $\psi = 0$ as x gets farther from the well.

In order to find the energies of these states, we make the following general observations:

1. For each state ψ_n, the number of nodes in the well region is $(n-1)$.

2. For small values of n, the finite well energies are close to the infinite well energies $E_n \approx -V_0 + \frac{p_n^2}{2m}$, with $p_n = \frac{\hbar}{\lambda_n} = \frac{n2\hbar}{L}$.

3. As n gets larger, and $E_n \to 0$, the states look less and less like the infinite well states, with more of ψ_n being outside of the well.

4. For unbound states, with $E > 0$, these are similar to traveling wave sinusoidal functions for $-\infty < x < +\infty$, with some alteration in the region near the well.

Building on this, we want to understand a more complicated system: the two-atom electron states. Then we now put two atoms **A** and **B**, a distance a apart from each other, modeling this as two finite wells of width L and depth $-V_0$, whose centers are a distance a apart from each other. Starting with $a \gg L$, the states of the two wells are almost unaffected by the other well. For each n, the electron is highly likely to be found near one atom or the other, which indicates there are two distinct single-electron states with identical energies: state has an electron almost certainly near atom **A**, and the other is such that the electron is almost certainly near atom **B**. When the atoms get closer together, the likelihood that the electron could be found in the region between the atoms is no longer ignorable, so we have to construct wave functions that combine probabilities of being near **A** with being near **B**. And the relative phases of the two atomic regions' contributions matters. When we combine independent wave functions into composite functions, N individual states lead to N linearly independent combined states. Then we can find the energy of electron in various situations:

1. The contribution of being near one atom center or the other is the Coulomb attraction of the nucleus on the electron, which we know gives $E \approx E_1$ for each state.

2. If the electron has any sizable probability to be found in the region between **A** and **B**, that is a negative charge between two positive atomic charge centers, giving an attractive force and a lower overall potential energy.

3. So, the “double-hump state” will have a slightly lower energy than the “up and down state” because of the higher electron density in between-atoms region in the first case.

4. Also, the “double-hump state”(see E_3 on the right of Fig 2.3.1) changes less rapidly than the state with normal sine or cosine waves (see E_2 on the right of Fig 2.3.1) as x changes, meaning that its $|\psi''|$ is smaller, giving it has ess kinetic energy.

The two possible states are known as “bonding” and “anti-bonding” molecular orbitals.

Then we can go to the N-atom electron states. If we have a 1D lattice of N atoms, modeled as a series of finite wells of width L and depth V_0, with a distance a from one well to the next, the combined states will again fall into sets roughly near the single-well energies. The N linearly independent wave functions will come from combining the states with all of the possible symmetric/anti-symmetric choices. At each value, $E \approx E_n$, there will be N distinct single-electron energies, spread out by a small deviations with $\Delta E_n \ll (E_{n+1} - E_n)$. We illustrate from the origin $n = 1$ single-atom states:

1. The lowest energy combination is the symmetric addition of the four contributions, $\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}$. This gives no additional nodes, and the non-zero probabilities in the regions between each atom reduces the overall energy.

2. Then we add them as $\mathbf{A} + \mathbf{B} - \mathbf{C} - \mathbf{D}$, giving one new node in the center of the pattern. The lower probability to have an electron in this **B** to **C** region increases the energy a bit compared the first case.

3. I now add them as $\mathbf{A} - \mathbf{B} - \mathbf{C} + \mathbf{D}$, giving two new nodes in the pattern. The lower probability to have an electron in the **A** to **B** and **C** to **D** regions increases the energy a bit compared the second case.

4. Then we add them anti-symmetrically as $\mathbf{A} - \mathbf{B} + \mathbf{C} - \mathbf{D}$, giving three new nodes in the pattern. The lower probability to have an electron in any the inter-atom regions increases

the energy a bit compared the other cases.

5. Going from the first case to the last, the ψ functions vary more rapidly with x, giving higher kinetic energy. This also contributes to the energy split between levels.

For the types of ridiculously large N-atom systems that make up condensed matter crystals, the end result is sets of energy levels which bunch together very closely. Within any bunch, the energies are so close together that for $N \approx 10^{20}$, it is reasonable to think of them as forming a continuous set of energies in a **band**. The fact that these bunches are widely spaced is what we call the **band gaps**.

2.3.1 1D Energy Band

Here I want to illustrate general conclusions in one dimension, that is where the occurrence rate of the twofold degeneracy is the greatest. When there are no interactions, the energy levels are just the Fig 2.3.2 (a). It is just a simple parabola.

Things are different when the curve nears the Bragg planes, in one dimension, Bragg planes are actually points. When q is near a Bragg plane with a corresponding reciprocal lattice vector K, the energy level becomes another free electron parabola centered at K, which indicates in Fig 2.3.2 (b). The degeneracy at the intersection point is split by $2|U_k|$, and both curves have zero slope at the intersection and then by taking this zero slope, we can redraw Fig 2.3.2 (b) into Fig 2.3.2 (c).

We then incorporate all these conditions in the original free electron case and we get Fig 2.3.2 (d). When we include all the Bragg planes and their associated Fourier components, we can get the set of curves shown in Fig 2.3.2 (e). And this method of demonstrating the energy levels is called the extended-zone scheme. If we want to include all the energy levels within the first Brillouin zone, then we have to translate Fig 2.3.2 (e) through reciprocal lattice vectors into the first Brillouin zone and hence we get Fig 2.3.2 (f) and it is called the reduced-zone scheme. Finally, if we want to repeat the energy levels in every Brillouin zone, then we can generate Fig

2.3.2 (g), and it is in a repeated-zone scheme. Fig 2.3.2 (g) emphasizes that a particular level at k can be described by any wave vector differing from k by a reciprocal lattice vector.

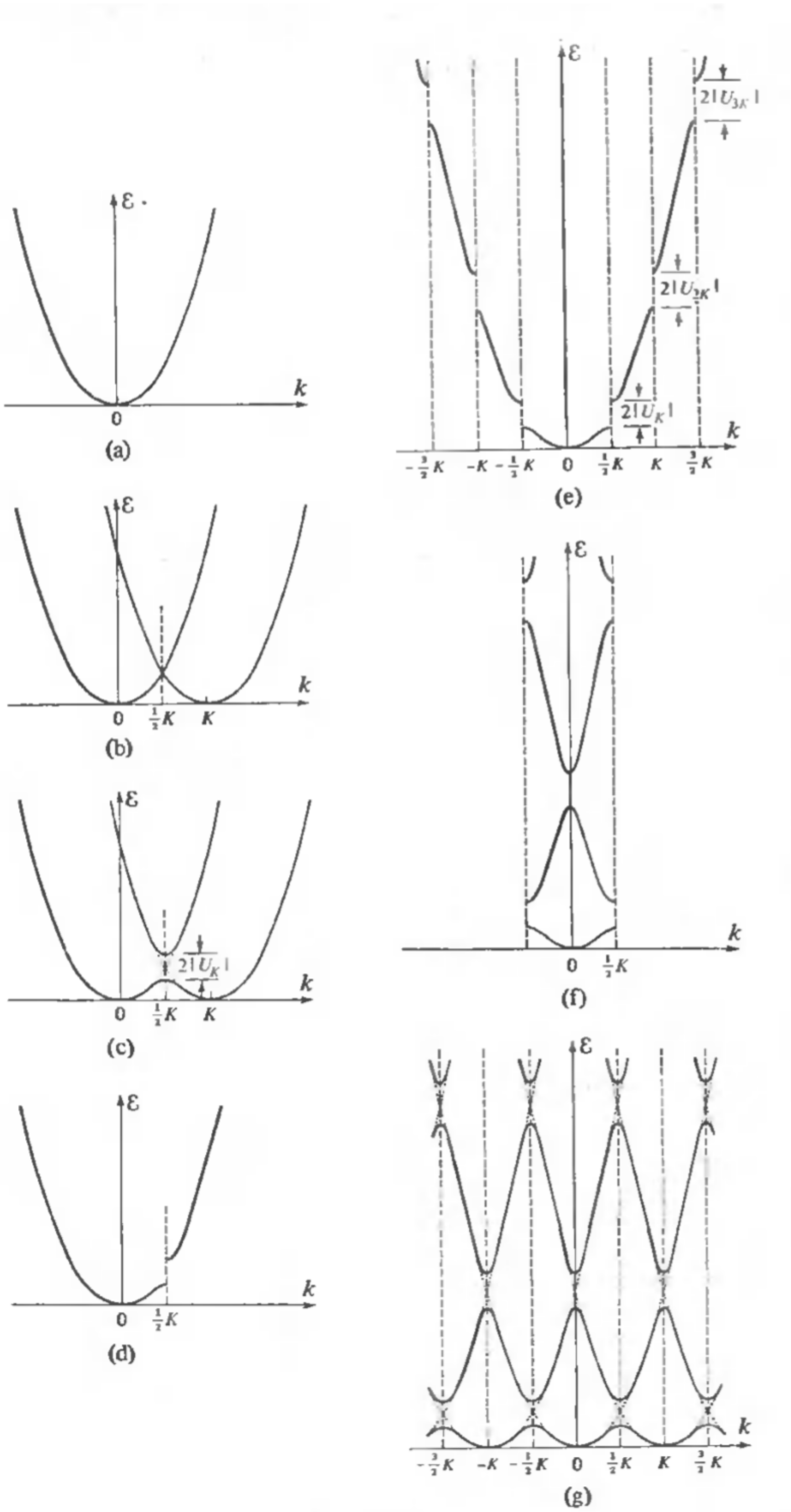


Figure 2.3.2. is from [1]. (a) The free electron case. (b) We determine the potential near Bragg plane. (c) We split the degeneracy at the point of intersection. (d) We add the degeneracy to the first free electron case. (e) Expand what we did to all the Bragg planes. (f) Constraining all the potentials to the first Brillouin zone. (g) Repeating to other Brillouin zones.

3
Potential and Models

3.1 Perturbation Theory

By using the known eigenvalues and eigenvectors of a known system, we can find small modifications to the energies and wave functions when the system is a little more complicated than the simple system. This is done by expressing the perturbation to the system in terms of an adjustable parameter, λ and then matching up terms of the same power of λ. "**Perturbation theory** is a systematic procedure for obtaining approximate solutions to the perturbed problem, by building on the known exact solutions to the unperturbed case."[2] We first find the first-order perturbed energy, i.e. the modification to E_n^0. We first introduce an equation:

$$H^0|\psi_n^1\rangle + H'|\psi_n^0\rangle = E_n^0|\psi_n^1\rangle + E_n^1|\psi_n^0\rangle. \tag{3.1.1}$$

And then we act the bra co-vector $\langle\psi_n^0|$ on 3.1.1, and it becomes:

$$\begin{aligned} \langle\psi_n^0|H^0|\psi_n^1\rangle + \langle\psi_n^0|H'|\psi_n^0\rangle &= \langle\psi_n^0|E_n^0|\psi_n^1\rangle + \langle\psi_n^0|E_n^1|\psi_n^0\rangle \\ E_n^0\langle\psi_n^0|\psi_n^1\rangle + \langle\psi_n^0|H'|\psi_n^0\rangle &= E_n^0\langle\psi_n^0|\psi_n^1\rangle + E_n^1\langle\psi_n^0|\psi_n^0\rangle \\ \langle\psi_n^0|H'|\psi_n^0\rangle &= E_n^1\langle\psi_n^0| = E_n^1 \cdot 1. \end{aligned}$$

So, we can find the first-order contribution to the energy E_n by evaluating the expectation value of the perturbing Hamiltonian on the n^{th} unperturbed wave function: $E_n^1 = \langle\psi_n^0|H'|\psi_n^0\rangle$.

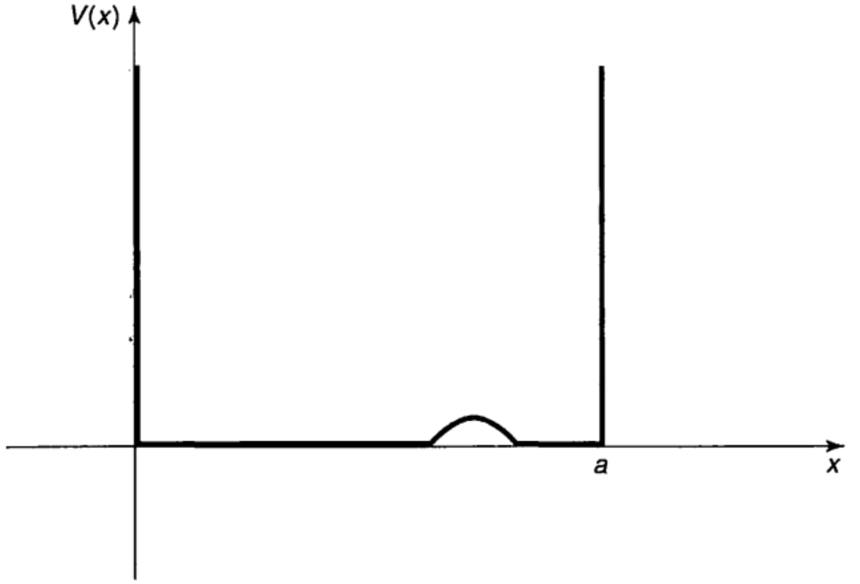


Figure 3.1.1. is from [2]. This is the perturbation in infinite square well.

This is the energy shift. Then we can discuss the first order wave function.

Working only to the first order, the perturbed wave function is expressed as the zeroth-order wave function, plus something different,

$$|\psi_n\rangle = |\psi_n^0\rangle + \lambda \cdot |\psi_n^1\rangle.$$

The additional term can be expanded in terms of the complete orthonormal basis provided by the eigenfunctions of the unperturbed Hamiltonian,

$$|\psi_n^1\rangle = \sum_{m \neq n} c_m^{(n)} |\psi_m^0\rangle. \tag{3.1.2}$$

We did not include the $m = n$ term because any contribution from ψ_n^0 could be incorporated into the first term in the equation above. As for the coefficients $c_m^{(n)}$, those are found using the orthonormality of the ψ_m^0 wave functions,

$$\langle\psi_m^0|\psi_n^1\rangle = \sum_{k \neq n} c_k^{(n)} \langle\psi_m^0|\psi_k^0\rangle = \sum_{k \neq n} c_k^{(n)} \delta_{mk} \Longrightarrow \langle\psi_m^0|\psi_n^1\rangle = c_m^{(n)}.$$

Going back to equation 3.1.1, we act on it with the bra co-vector $\langle\psi_m^0|$, this time with $m \neq n$.

$$
\begin{aligned}
H^0|\psi_n^1\rangle + H'|\psi_n^0\rangle &= E_n^0|\psi_n^1\rangle + E_n^1|\psi_n^0\rangle \\
\langle\psi_m^0|H^0|\psi_n^1\rangle + \langle\psi_m^0|H'|\psi_n^0\rangle &= \langle\psi_m^0|E_n^0|\psi_n^1\rangle + \langle\psi_m^0|E_n^1|\psi_n^0\rangle \\
E_m^0\langle\psi_m^0|\psi_n^1\rangle + \langle\psi_m^0|H'|\psi_n^0\rangle &= E_n^0\langle\psi_m^0|\psi_n^1\rangle + E_n^1\langle\psi_m^0|\psi_n^0\rangle \\
E_m^0\langle\psi_m^0|\psi_n^1\rangle + \langle\psi_m^0|H'|\psi_n^0\rangle &= E_n^0\langle\psi_m^0|\psi_n^1\rangle + E_n^1 \cdot 0.
\end{aligned}
$$

The last term is zero because of the orthogonality of the ψ_n^0 set.

Then I want to introduce the Second-Order Energies. Here we take the inner product of the second order of 3.1.1 with ψ_n^0:

$$\langle\psi_n^0\,|\,H^0\,\psi_n^2\rangle + \langle\psi_n^0\,|\,H'\,\psi_n^1\rangle = E_n^0\langle\psi_n^0\,|\,\psi_n^2\rangle + E_n^1\langle\psi_n^0\,|\,\psi_n^1\rangle + E_n^2\langle\psi_n^0\,|\,\psi_n^0\rangle.$$

Since H^0 has hermiticity:

$$\langle\psi_n^0\,|\,H^0\,\psi_n^2\rangle = \langle H^0\,\psi_n^0\,|\,\psi_n^2\rangle = E_n^0\langle\psi_n^0\,|\,\psi_n^2\rangle$$

Since $\langle\psi_n^0\,|\,H^0\,\psi_n^2\rangle + \langle = E_n^0\langle\psi_n^0\,|\,\psi_n^2\rangle$ and $\langle\psi_n^0\,|\,\psi_n^0\rangle = 1$, then we can develop an equation for E_n^2:

$$E_n^2 = \langle\psi_n^0\,|\,H'|\psi_n^1\rangle - E_n^1\langle\psi_n^0\,|\,\psi_n^1\rangle.$$

And we know what is $\langle\psi_n^0\,|\,\psi_n^1\rangle$:

$$\langle\psi_n^0\,|\,\psi_n^1\rangle = \sum_{m\neq n} c_m^{(n)}\langle\psi_n^0\,|\,\psi_n^0\rangle = 0.$$

Put this back to the E_n^2 equation, we get:

$$E_n^2 = \langle\psi_n^0\,|\,H'|\psi_n^1\rangle = \sum_{m\neq n} c_m^{(n)}\langle\psi_n^0\,|H'|\psi_n^0\rangle = \sum_{m\neq n} \frac{\langle\psi_m^0\,|\,H'|\psi_n^0\rangle\langle\psi_n^0\,|\,H'|\psi_m^0\rangle}{E_n^0 - E_m^0},$$

and finally we get the second order energy to be:

$$E_n^2 = \sum_{m\neq n} \frac{|\langle\psi_m^0\,|\,H'|\psi_n^0\rangle|^2}{E_n^0 - E_m^0}. \tag{3.1.3}$$

3.2 Weak Potential

Starting from the simple system of plane wave electron motions, we introduce the perturbation is periodic potential of the atoms in the lattice. Now let's consider the general approach to the Schrödinger's Equation for weak potentials. This following description follows Ashcroft and Mermin[1]. Recall from Bloch's Theorem with $\mathbf{k}$ the crystal momentum, we have:

$$\psi_{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{K}} c_{\mathbf{k}-\mathbf{K}}\, e^{i(\mathbf{k}-\mathbf{K})\cdot\mathbf{r}} \tag{3.2.1}$$

and recall from the proof of Bloch's Theorem:

$$\left(\frac{\hbar^2}{2m}(\mathbf{k}-\mathbf{K})^2 - \mathcal{E}\right) c_{\mathbf{k}-\mathbf{K}} + \sum_{\mathbf{K}'} U_{\mathbf{K}'-\mathbf{K}}\, c_{\mathbf{q}-\mathbf{K}'} = 0. \tag{3.2.2}$$

Since $\mathbf{k}$ has many solutions lying in the first Brillouin zone that correspond with each reciprocal lattice, if we consider the free electron case, then the previous equation becomes:

$$(\mathcal{E}^0_{\mathbf{k}-\mathbf{K}} - \mathcal{E})c_{\mathbf{k}-\mathbf{K}} = 0. \tag{3.2.3}$$

In order to make the equation simpler, we set:

$$\mathcal{E}^0_q = \frac{\hbar^2}{2m}q^2.$$

To get 3.2.3 to work, we require either $c_{\mathbf{k}-\mathbf{K}} = 0$ or $\mathcal{E} = \mathcal{E}^0_{\mathbf{k}-\mathbf{K}}$. Therefore, we divide the treatment of the weak potential into two cases.

1. The first case is to fix $\mathbf{k}$ and take a specific reciprocal lattice vector $\mathbf{K}_1$, so the free electron energy $\mathcal{E}^0_{k-K_1}$ is greater than the $\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}$ by the energy needed to the next state U, that is:

$$|\mathcal{E}^0_{\mathbf{k}-\mathbf{K}_1} - \mathcal{E}^0_{\mathbf{k}-\mathbf{K}}| \gg U, \tag{3.2.4}$$

for fixed $\mathbf{k}$ and all $\mathbf{K} \neq \mathbf{K}_1$.

If we let $\mathbf{K} \neq \mathbf{K}_1$, then 3.2.4 will become:

$$(\mathcal{E} - \mathcal{E}^0_{\mathbf{k}-\mathbf{K}_1})c_{\mathbf{k}-\mathbf{K}_1} = \sum_{\mathbf{K}} U_{\mathbf{k}-\mathbf{K}_1} c_{\mathbf{k}-\mathbf{K}} \tag{3.2.5}$$

From the proof of Bloch's Theorem, we have that when $\mathbf{K}=0$, then $U_{\mathbf{K}}=0$. Other terms from the right hand side of the previous equation will vanish, until we reach $\mathbf{K}=\mathbf{K}_1$. Therefore, the sum will only be left with second order in U:

$$c_{\mathbf{k}-\mathbf{K}}=\frac{U_{\mathbf{K}_1-\mathbf{K}}c_{\mathbf{k}-\mathbf{K}_1}}{\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}}+\sum_{\mathbf{K}\neq\mathbf{K}_1}\frac{U_{\mathbf{K'}-\mathbf{K}}c_{\mathbf{k}-\mathbf{K'}}}{\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}}.$$

Since there is no near degeneracy, we have:

$$c_{\mathbf{k}-\mathbf{K}}=\frac{U_{\mathbf{K}_1-\mathbf{K}}c_{\mathbf{k}-\mathbf{K}_1}}{\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}}+\mathbf{O}(\mathbf{U}^2).$$

Put this equation back to 3.2.5, we have:

$$(\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}_1})c_{\mathbf{k}-\mathbf{K}_1}=\sum_{\mathbf{K}}\frac{U_{\mathbf{K}-\mathbf{K}_1}U_{\mathbf{K}_1-\mathbf{K}}}{\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}}c_{\mathbf{k}-\mathbf{K}_1}+\mathbf{O}(\mathbf{U}^3).$$

Therefore the perturbed energy level $\mathcal{E}$ is greater than the free electron value by order of U^2. To solve the previous equation for $\mathcal{E}$, we have to change the index of the denominator from the right hand side:

$$\mathcal{E}=\mathcal{E}^0_{\mathbf{k}-\mathbf{K}_1}+\sum_{\mathbf{K}}\frac{|U_{\mathbf{K}-\mathbf{K}_1}|^2}{\mathcal{E}^0_{\mathbf{k}-\mathbf{K}_1}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}}+\mathbf{O}(\mathbf{U}^3).$$

2. Then we will discuss the second case. That is we assume the value of $\mathbf{k}$ such that the reciprocal lattice vectors $\mathbf{K_1}$,...,$\mathbf{K_m}$ and the $\mathcal{E}^0_{\mathbf{k}-\mathbf{K_1}},...,\mathcal{E}^0_{\mathbf{k}-\mathbf{K_m}}$ are all within order of U of each other, but they are far away from $\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}$ by the scale of U:

$$|\mathcal{E}^0_{\mathbf{k}-\mathbf{K}}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K_i}}|\gg U,$$

with $i=1,...,m$, and $\mathbf{K}\neq\mathbf{K_1},...,\mathbf{K_m}$.

In this case we have to separate those equations given by 3.2.2 when we let $\mathbf{K}$ equal to any of the m values $\mathbf{K_1},...,\mathbf{K_m}$. This gives m equations relating to the single equation in 3.2.5 in the non-degenerate case. When we separate the m equations from the sum that contain coefficients $c_{\mathbf{k}-\mathbf{K_j}}$ with $j=1,...,m$ will not be small when the limit vanishes from the remaining $c_{\mathbf{k}-\mathbf{K}}$, which will be at most of order U. Then we have:

$$(\mathcal{E}-\mathcal{E}^0_{\mathbf{k}-\mathbf{K_i}})c_{\mathbf{k}-\mathbf{K_i}}=\sum_{j=1}^{m}U_{\mathbf{K_j}-\mathbf{K_i}}c_{\mathbf{k}-\mathbf{K_j}}+\sum_{\mathbf{K}\neq\mathbf{K_1},...,\mathbf{K_m}}U_{\mathbf{K}-\mathbf{K_i}}c_{\mathbf{k}-\mathbf{K}}, \tag{3.2.6}$$

with $i = 1, ..., m$.

Then we follow the same separating method as the sum, we then write 3.2.2 for the remaining levels:

$$c_{\mathbf{k-K}} = \frac{1}{\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K}}} \left(\sum_{j=1}^{m} U_{\mathbf{K_j-K_i}} c_{\mathbf{k-K_j}} + \sum_{\mathbf{K \neq K_1,...,K_m}} U_{\mathbf{K'-K_i}} c_{\mathbf{k-K'}} \right),$$

with $\mathbf{K} \neq \mathbf{K_1}$,...,$\mathbf{K_m}$.

Since $c_{\mathbf{k-K}}$ will be most of order U when $\mathbf{K} \neq \mathbf{K_1}$,...,$\mathbf{K_m}$, the previous equation gives:

$$c_{\mathbf{k-K}} = \frac{1}{\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K}}} \sum_{j=1}^{m} U_{\mathbf{K_j-K_i}} c_{\mathbf{k-K_j}} + \mathbf{O}(\mathbf{U}^2).$$

Putting this equation back to Eqn.4.2.6, we get:

$$\left(\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K_i}}\right) c_{\mathbf{k-K_i}} = \sum_{j=1}^{m} U_{\mathbf{K_j-K_i}} c_{\mathbf{k-K_j}} + \sum_{j=1}^{m} \left(\sum_{\mathbf{K \neq K_1,...,K_m}} \frac{U_{\mathbf{K-K_i}} U_{\mathbf{K_j-K}}}{\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K}}} \right) c_{\mathbf{k-K_j}} + \mathbf{O}(\mathbf{U}^3). \tag{3.2.7}$$

Compare this equation with what we got from Case 1: $(\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K}_1}) c_{\mathbf{k-K}_1} = \sum_{\mathbf{K}} \frac{U_{\mathbf{K-K}_1} U_{\mathbf{K}_1\mathbf{-K}}}{\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K}}} c_{\mathbf{k-K}_1} + \mathbf{O}(\mathbf{U}^3)$, where there is no near degeneracy. And we found an explicit expression for the shift in energy of order U^2. Nevertheless, we find that to accuracy of order U^2 the determination of the shifts in the m nearly degeneracy levels reduces to the solution of m coupled equations for the $c_{\mathbf{k-K_j}}$. The coefficients in the second term on the right-hand side of these equations are of higher order in U than those in the first. Then we replace 3.2.7 by a sinple equation:

$$\left(\mathcal{E} - \mathcal{E}^0_{\mathbf{k-K_i}}\right) c_{\mathbf{k-K_i}} = \sum_{j=1}^{m} U_{\mathbf{K_j-K_i}} c_{\mathbf{k-K_j}},$$

with $i = 1, ..., m$.

These are the general equations for a system of m quantum levels.

3.3 Tight-binding Model

In the previous section, we discussed the weakly perturbed periodic potential and now let us explore a totally different model that is more suitable to graphene, that is the tight-binding model. In this model, we are treating the solid (graphene) as a group of weakly interactive neutral atoms. “The tight-binding approximation deals with the case in which the overlap of atomic wave function is enough to require corrections to the picture of isolated atoms, but not so much as to render the atomic description completely irrelevant. The approximation is most useful for describing the energy bands that arise from the partially filled *d*-shells of transition metal atoms and for describing the electronic structure of insulators."[1] We know from the previous chapters that graphene is a semi-metal, which means at low temperature it has the property of an insulator. Hence, the tight-binding model can fit graphene pretty well.

Now I am going to introduce the general formulation of the tight-binding approximation. This introduction follows Chapter 10 in Ashcroft and Mermin's *Solid State Physics.*[1] To develop the tight-binding approximation, we assume that in the neighborhood of each lattice point the periodic crystal Hamiltonian, H, can be estimated by the Hamiltonian, H_{at}, of a single atom located at each lattice point. "We also assumed the bound levels of atomic Hamiltonian are well localized; i.e., if ψ_n is a bound level of H_{at} for an atom at the origin."[1]

$$H_{at}\psi_n = E_n\psi_n. \tag{3.3.1}$$

Then we restrict $\psi_n(\mathbf{r})$ to be very small when r is large in comparison to the lattice constant $|\mathbf{R}|$. If r is comparable different to $|\mathbf{R}|$, then H is no longer equal to H_{at}. Then the wave function $\psi_n(\mathbf{r})$ will become a good estimation to the stationary-state wave function for the full Hamiltonian, which has the eigenvalue E_n. And we can also know the wave functions $\psi_n(\mathbf{r}-\mathbf{R})$, when a set of $\mathbf{R}$ are in the Bravais lattice and we know H has periodicity over the lattice. The main idea is to set up the Perturbation theory when the wave functions are perturbed when r is different from $|\mathbf{R}|$. In this extreme case, we rewrite the crystal Hamiltonian as:

$$H = H_{at} + \Delta U(\mathbf{r}).$$

Here $\Delta U(\mathbf{r})$ represents the corrections with all the atomic potential that can construct the full periodic potential of the lattice. From the graph, we can see that when $\psi_n(\mathbf{r})$ is at the discontinuity, we have $\Delta U(\mathbf{r})$ to make up that point. The $\Delta U(\mathbf{r})$ line lies along the line of atomic sites. When the upper curve gets larger, the lower curve $\Delta U(\mathbf{r})$ gets smaller and vice

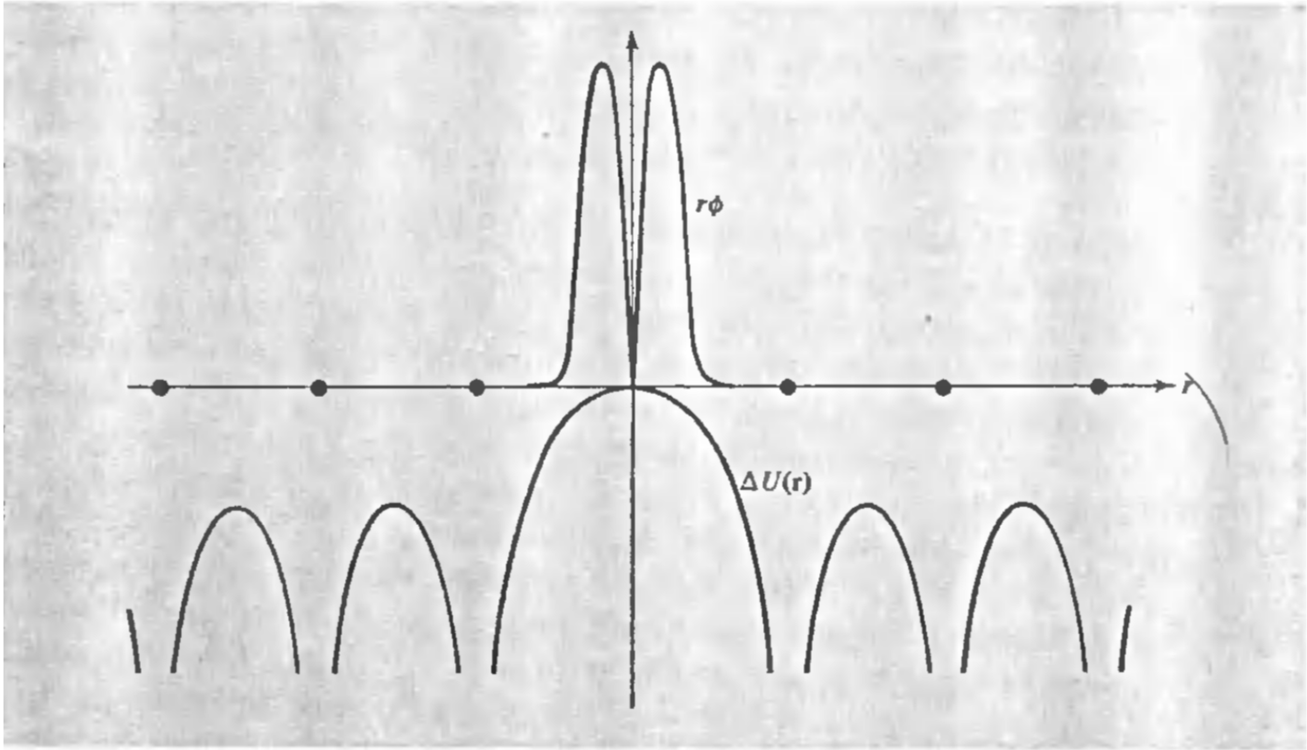


Figure 3.3.1. is from [1]. This figure explains how $\Delta U(\mathbf{r})$ corrects the discontinuity in $\psi_n(\mathbf{r})$.

versa. Hence we define a new wave function $\psi_n(\mathbf{r}-\mathbf{R})$ corresponding with N levels in the periodic potential, for N sites $\mathbf{R}$ in the lattice. We must obey Bloch's theorem, so that we need to find N linear combinations of the degenerate wave functions which are satisfying the Bloch condition 2.2.2:

$$\psi_n(\mathbf{r}+\mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}\psi_n(\mathbf{r}). \tag{3.3.2}$$

Then we sum up all N linear combinations:

$$\psi_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}}\psi_n(\mathbf{r}-\mathbf{R}), \tag{3.3.3}$$

with the set of $\mathbf{k}$ in the first Brillouin zone and it satisfies the Born-von Karman boundary condition. To prove the wave function $\psi_n(\mathbf{r}+\mathbf{R})$ satisfies the Bloch's theorem, we have:

$$\begin{aligned}\psi_n(\mathbf{r}+\mathbf{R}) &= \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}'}\psi_n(\mathbf{r}+\mathbf{R}-\mathbf{R}') \\ &= e^{i\mathbf{k}\cdot\mathbf{R}}\left[\sum_{\mathbf{R}'} e^{i\mathbf{k}\cdot(\mathbf{R}'-\mathbf{R})}\psi_n(\mathbf{r}+\mathbf{R}-\mathbf{R}')\right] \\ &= e^{i\mathbf{k}\cdot\mathbf{R}}\left[\sum_{\overline{\mathbf{R}}} e^{i\mathbf{k}\cdot\overline{\mathbf{R}}}\psi_n(\mathbf{r}\}-\overline{\mathbf{R}})\right] \\ &= e^{i\mathbf{k}\cdot\mathbf{R}}\psi(\mathbf{r}).\end{aligned}$$

Hence, we can conclude that the set of wave functions does fit the Bloch's condition with the set of $\mathbf{k}$s. The energy bands $\mathcal{E}$ are just the energy E_n at the atomic level. Then we define a solution of the full Schrödinger's equation that still obeys the general form of $\psi_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}}\psi_n(\mathbf{r}-\mathbf{R})$:

$$\psi_n(\mathbf{r}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}}\phi(\mathbf{r}-\mathbf{R}). \tag{3.3.4}$$

Then the next step would be to define what is $\phi(\mathbf{r}-\mathbf{R})$. Since we expect $\phi(\mathbf{r})$ to be close to the degenerate wave functions $\psi_n(\mathbf{r})$, we have:

$$\phi(\mathbf{r}) = \sum_n b_n\psi_n(\mathbf{r}). \tag{3.3.5}$$

Then we multiply the crystal Schrödinger's equation by the wave function $\psi_m^*(\mathbf{r})$ and integrate over the set if $\mathbf{r}$ we get:

$$H\psi(\mathbf{r}) = (H_{at}+\Delta U(\mathbf{r}))\psi(\mathbf{r}) = \mathcal{E}(\mathbf{k})\psi(\mathbf{r}) \tag{3.3.6}$$

$$\int \psi_m^*(\mathbf{r})H_{at}\psi(\mathbf{r})\,d\mathbf{r} = \int (H_{at}\psi_m(\mathbf{r}))^*\psi(\mathbf{r})\,d\mathbf{r} = E_m\int \psi_m^*(\mathbf{r})\psi(\mathbf{r})\,d\mathbf{r}, \tag{3.3.7}$$

and then we find:

$$(\mathcal{E}(\mathbf{k})-E_m)\int \psi_m^*(\mathbf{r})\psi(\mathbf{r})\,d\mathbf{r} = \int \psi_m^*(\mathbf{r})\Delta U(\mathbf{r})\psi(\mathbf{r})\,d\mathbf{r}. \tag{3.3.8}$$

Now we put 3.3.4 and 3.3.5 into 3.3.8 and combining the knowledge of orthonormality of the atomic wave functions. we get:

$$\int \psi_m^*(\mathbf{r})\psi(\mathbf{r})\,d\mathbf{r} = \delta_{nm}.$$

Then we can construct an eigenvalue equation that can help us to find the coefficients $b_n(\mathbf{k})$ and the Bloch energies $\mathcal{E}(\mathbf{k})$:

$$\begin{aligned}(\mathcal{E}(\mathbf{k}) - E_m)b_m = -(\mathcal{E}(\mathbf{k}) - E_m)\sum_n &\left(\sum_{\mathbf{R}\neq 0}\int \psi_m^*(\mathbf{r})\psi_n(\mathbf{r}-\mathbf{R})e^{i\mathbf{k}\cdot\mathbf{R}}\,d\mathbf{r}\right)b_n \\ &+\sum_n\left(\int \psi_m^*(\mathbf{r})\Delta U(\mathbf{r})\psi_n(\mathbf{r})\,d\mathbf{r}\right)b_n \\ &+\sum_n\left(\sum_{\mathbf{R}\neq 0}\int \psi_m^*(\mathbf{r})\Delta U(\mathbf{r})\psi_n(\mathbf{r}-\mathbf{R})e^{i\mathbf{k}\cdot\mathbf{R}}\,d\mathbf{r}\right)b_n\end{aligned}$$

The first term on the right of the previous equation has the form:

$$\int d\mathbf{r}\,\psi_m^*(\mathbf{r})\,\psi_n(\mathbf{r}-\mathbf{R}),$$

is known as the overlap integral and the tight-binding approximation assume it is small compare to unity. We also assume the third term to be small because it also contains two wave functions evaluated at different locations. The second term is also small because we assume before that $\psi_n(\mathbf{r})$ is very small when r is different to the lattice constant $|\mathbf{R}|$.

Consequently, the right hand side of the previous equation is always small. For the left hand side, it is either $\mathcal{E}(\mathbf{k}) - E_m$t is small or b_m is small. Therefore we can conclude that:

$$\mathcal{E}(\mathbf{k}) \approx E_0, b_m \approx 0 \text{ unless } E_m \approx E_0. \tag{3.3.9}$$

3.3.1 Tight-binding Model for Graphene

Now, let us explore how the tight-binding model applied to graphene, that is to apply the tight-binding model on the honeycomb lattice with two atoms per unit cell.

On the left, Fig 3.3.2 indicates the honeycomb lattice of graphene. The $\mathbf{a}_1$ and $\mathbf{a}_2$ are the unit vectors. On the right it is the first Brillouin zone of graphene. The Dirac points are at $\mathbf{K}$ and $\mathbf{K}'$ points.

The following analysis is followed by M. O. Goerbig[5]. We first write a wave function for two atoms in the unit cell:

$$\psi_{\mathbf{k}}(\mathbf{r}) = a_{\mathbf{k}}\psi_{\mathbf{k}}^{(A)}(\mathbf{r}) + b_{\mathbf{k}}\psi_{\mathbf{k}}^{(B)}(\mathbf{r}), \tag{3.3.10}$$

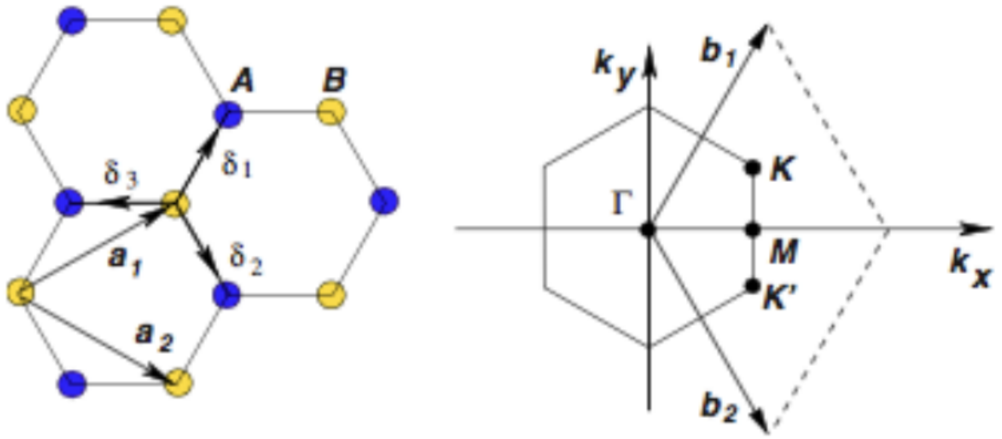


Figure 3.3.2. is from [11]. This is the honeycomb lattice with the triangular Bravais lattice.

with $a_{\mathbf{k}}$ and $b_{\mathbf{k}}$ are complex functions of the set of $\mathbf{k}$. From the general case, we know both $\psi_{\mathbf{k}}^{(A)}(\mathbf{r})$ and $\psi_{\mathbf{k}}^{(B)}(\mathbf{r})$ satisfy the Bloch condition with:

$$\psi_{\mathbf{k}}^{(j)}(\mathbf{r}) = \sum_{\mathbf{R}_l} e^{i\mathbf{k}\cdot\mathbf{R}_l}\,\phi^{(j)}(\mathbf{r}+\boldsymbol{\delta}_j-\mathbf{R}_l),$$

with j labels the order of the two atoms within the unit cell and $\boldsymbol{\delta}_j$ is a vector that connects the atom with the Bravais lattice. The function ϕ here are similar to the general case, which are the atomic orbital wave functions. And then again, we want to find the solutions to the Schrödinger's equation. And like in the general case, we multiply the each side of the Schrödinger's equation with $\psi_{\mathbf{k}}^*$. Then we define the Hamiltonian in the matrix way:

$$\mathcal{H}_{\mathbf{k}} \equiv \begin{pmatrix} \psi_{\mathbf{k}}^{(A)}(\mathbf{r})^* H \psi_{\mathbf{k}}^{(A)}(\mathbf{r}) & \psi_{\mathbf{k}}^{(A)}(\mathbf{r})^* H \psi_{\mathbf{k}}^{(B)}(\mathbf{r}) \\ \psi_{\mathbf{k}}^{(B)}(\mathbf{r})^* H \psi_{\mathbf{k}}^{(A)}(\mathbf{r}) & \psi_{\mathbf{k}}^{(B)}(\mathbf{r})^* H \psi_{\mathbf{k}}^{(B)}(\mathbf{r}) \end{pmatrix} \equiv \mathcal{H}_{\mathbf{k}}^{\dagger}. \tag{3.3.11}$$

Then we multiply the Hamiltonian matrix with $\psi_{\mathbf{k}}^*$:

$$(a_{\mathbf{k}}^*, b_{\mathbf{k}}^*)\mathcal{H}_{\mathbf{k}} \begin{pmatrix} a_{\mathbf{k}} \\ b_{\mathbf{k}} \end{pmatrix} = \epsilon_{\mathbf{k}}(a_{\mathbf{k}}^*, b_{\mathbf{k}}^*)\mathcal{S} \begin{pmatrix} a_{\mathbf{k}} \\ b_{\mathbf{k}} \end{pmatrix},$$

with the overlap matrix $\mathcal{S}$ defined by:

$$\mathcal{S}_{\mathbf{k}} \equiv \begin{pmatrix} \psi_{\mathbf{k}}^{(A)}(\mathbf{r})^* \psi_{\mathbf{k}}^{(A)}(\mathbf{r}) & \psi_{\mathbf{k}}^{(A)}(\mathbf{r})^* \psi_{\mathbf{k}}^{(B)}(\mathbf{r}) \\ \psi_{\mathbf{k}}^{(B)}(\mathbf{r})^* \psi_{\mathbf{k}}^{(A)}(\mathbf{r}) & \psi_{\mathbf{k}}^{(B)}(\mathbf{r})^* \psi_{\mathbf{k}}^{(B)}(\mathbf{r}) \end{pmatrix} \equiv \mathcal{S}_{\mathbf{k}}^{\dagger}.$$

The overlap matrix $\mathcal{S}$ covers the nonorthogonality of the atomic wave functions. To find the eigenvalues $\epsilon_{\mathbf{k}}$, we determine:

$$\det = [\mathcal{H}_{\mathbf{k}} - \epsilon_{\mathbf{k}}^{\lambda}\mathcal{S}_{\mathbf{k}}] = 0,$$

with λ being the energy bands and the number of energy bands equal to the number of solutions to the secular equation. Now we zoom in on the honeycomb lattice.

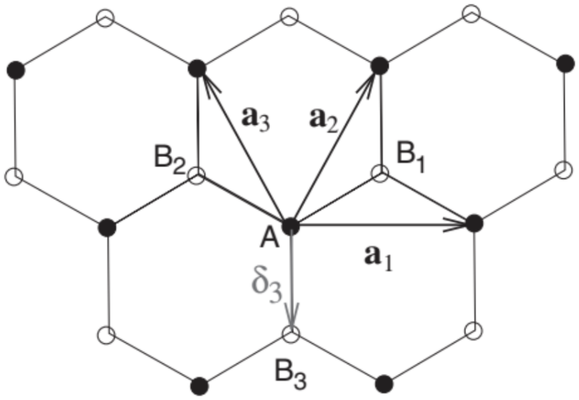


Figure 3.3.3. is from [5]. This is the honeycomb lattice with two sublattices.

In Fig 3.3.3, the site we choose to evaluate the the tight-binding model is in the sublattice A. We are considering the three Bravais lattice: $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$. We define the nearest neighbor(NN) hopping amplitude to be:

$$t \equiv \int d^2r \phi^{A*}(\mathbf{r})\Delta V\, \phi^{B}(\mathbf{r}+\boldsymbol{\delta}_3). \tag{3.3.12}$$

The hopping amplitude is the distance when you "hop" on to the next nearest neighbor(NNN) and in Fig 3.3.3 we hop to the next nearest neighbor through $\mathbf{a}_1$. Hence we define:

$$t_{\mathrm{NNN}} \equiv \int d^2r \phi^{A*}(\mathbf{r})\, \phi^{B}(\mathbf{r}+\mathbf{a}_1).$$

Then we normalize the atomic wave function and get:

$$\int d^2r \phi^{(j)*}(\mathbf{r})\phi^{(j)}(\mathbf{r}) = 1.$$

We then define the overlap corrections to be:

$$s \equiv \int d^2r \phi^{A*}(\mathbf{r})\phi^{B}(\mathbf{r}+\boldsymbol{\delta}_3).$$

We only consider the overlap between orbitals of nearest neighbor and next nearest neighbor. For instance, in Fig 3.3.3, we move from sides B_1 to B_2 by shifts through the lattice vector $\mathbf{a}_2$ and we have $\mathbf{a}_3 \equiv \mathbf{a}_2 - \mathbf{a}_1$. The next step is to consider the phase factors and we sum them of the nearest neighbor as:

$$\gamma_{\mathbf{k}}^{AA} \equiv 1 + e^{i\mathbf{k}\cdot\mathbf{a}_2} + e^{i\mathbf{k}\cdot\mathbf{a}_3}.$$

Just as we construct the Hamiltonian matrix, we define the diagonal elements of the hopping matrix first:

$$t_{\mathbf{k}}^{AA} = t_{\mathbf{k}}^{BB} = 2t_{\mathrm{NNN}} \sum_{i=1}^{3} \cos(\mathbf{k}\cdot\mathbf{a}_i) = t_{\mathrm{NNN}}(|\gamma_{\mathbf{t}}|^2 - 3). \tag{3.3.13}$$

Then we find the determinant as:

$$\det = \begin{bmatrix} t_{\mathbf{k}}^{AA} - \boldsymbol{\epsilon}_{\mathbf{k}} & (t - s\boldsymbol{\epsilon}_{\mathbf{k}})\gamma_{\mathbf{t}}^{*} \\ (t - s\boldsymbol{\epsilon}_{\mathbf{k}})\gamma_{\mathbf{t}} & t_{\mathbf{k}}^{AA} - \boldsymbol{\epsilon}_{\mathbf{k}} \end{bmatrix}, \tag{3.3.14}$$

with two solutions:

$$\boldsymbol{\epsilon}_{\mathbf{k}}^{\lambda} = \frac{t_{\mathbf{k}}^{AA} \pm \lambda t|\gamma_{\mathbf{t}}|}{1 \pm \lambda s|\gamma_{\mathbf{t}}|}. \tag{3.3.15}$$

We now assume $s \ll 1$ and $t_{\mathrm{NNN}} \ll t$ and the previous equation becomes:

$$\begin{aligned} \boldsymbol{\epsilon}_{\mathbf{k}}^{\lambda} \simeq& t_{\mathbf{k}}^{AA} + \lambda t|\gamma_{\mathbf{t}}| - st|\gamma_{\mathbf{t}}|^2 = t_{\mathrm{NNN}}|\gamma_{\mathbf{t}}|^2 + \lambda t|\gamma_{\mathbf{t}}| \\ &= t'_{\mathrm{NNN}}\left[3 + 2\sum_{i=1}^{3}\cos(\mathbf{k}\cdot\mathbf{a}_i)\right] + \lambda t\sqrt{3 + 2\sum_{i=1}^{3}\cos(\mathbf{k}\cdot\mathbf{a}_i)}. \end{aligned}$$

Therefore we have found the eigenvalues of the hopping matrix and hence we know the energy level of graphene.

4
Hamiltonian Matrix

4.1 Quantum Operators

This project is heavily reliant on quantum mechanics, especially its use of the operators. Since graphene is symmetric and has periodic wave functions, we want to explore the translational operator. The translational operator takes a function and shifts it a distance a. The operator that accomplishes this is defined by

$$\begin{aligned}\hat{T}(a)\psi(x) &= \psi'(x) = \psi(x-a) \qquad (4.1.1)\\ &= \sum_{m=0}^{\infty}\frac{1}{n!}(-a)^n\frac{d^n}{dx^n}\psi(x)\\ &= \sum_{m=0}^{\infty}\frac{1}{n!}\left(\frac{-ia}{\hbar}\hat{P}\right)^n\psi(x). \qquad (4.1.2)\end{aligned}$$

Therefore

$$\hat{T}(a) = e^{\frac{-ia}{\hbar}\hat{P}}, \qquad (4.1.3)$$

$$\hat{T}(a)^{-1} = \hat{T}(a) = \hat{T}(a)^{\dagger}. \qquad (4.1.4)$$

Here we can say that momentum is the "generator" of translations. Then we define another operator $\hat{Q}'$ called the transformed operator and it is defined by

$$\langle\psi'|\hat{Q}|\psi'\rangle = \langle\psi|\hat{Q}'|\psi\rangle. \qquad (4.1.5)$$

This equation means the transformed operator $\hat{Q}'$ gives the same expected value when acts on untranslated state ψ as the untransformed operator $\hat{Q}$ acts on translated state ψ'. Therefore $\hat{Q}' = \hat{T}^\dagger \hat{Q}\hat{T}$. Use the "bra-ket" notation, we have $\hat{T}|f\rangle \equiv |Tf\rangle$, where f is a wave function and $\langle Tf| = \langle f|\hat{T}^\dagger$.

Then let us discuss about translational symmetry. A system is translational invariant (it has translational symmetry) if the Hamiltonian is unchanged by the transformation:

$$\hat{H}' = \hat{T}^\dagger \hat{H}\hat{T}, \tag{4.1.6}$$

$$\text{and}\, \hat{H}\hat{T} = \hat{T}\hat{H}. \tag{4.1.7}$$

To talk about a system, it has translational symmetry if the Hamiltonian commutes with the translational operator:

$$[\hat{H}, \hat{T}] = 0. \tag{4.1.8}$$

And if $V(x+a) = V(x)$ holds for continuous sets of a, we called continuous symmetry. However, graphene has discrete translational symmetry and we will discuss this more when we see how to construct the Hamiltonian matrix.

4.2 Linear Algebra and Hamiltonian Matrix

To approach quantum mechanics through linear algebra, we need to first consider operators in matrix mechanics. Now, let's introduce a simple one-body potential that satisfies periodic boundary conditions: $\psi(x + a) = \psi(x)$. The plane-wave basis we choose to work on is defined by:

$$\psi_n^{(0)}(x) = \sqrt{\frac{1}{a}} e^{(i\frac{2\pi n}{a}x)}. \tag{4.2.1}$$

The kinetic energy operator is: $\hat{K} = \frac{\hat{P}^2}{2m}$, where $\hat{P}$ is the momentum operator. We act the kinetic energy operator on the plane-wave basis, we get:

$$\hat{K}\psi_n^{(0)}(x) = \frac{\hat{P}}{2m}\left[-i\hbar\left(\sqrt{\frac{1}{a}} i \frac{2\pi n}{a} e^{(i\frac{2\pi n}{a}x)}\right)\right] \tag{4.2.2}$$

$$= \frac{\hbar^2}{2m}\sqrt{\frac{1}{a}} \frac{4\pi^2 n^2}{a^2} e^{(i\frac{2\pi n}{a}x)}, \tag{4.2.3}$$

with eigenvalue $4\left(\frac{\hbar^2\pi^2n^2}{2ma^2}\right)$. We first define the Hamiltonian:

$$H_{nm} = \langle\psi_n^{(0)}|(H_0 + V(x))|\psi_m^{(0)}\rangle.$$

Then we write this equation in terms of a matrix:

$$\sum_{m=1}^{\infty} H_{nm}c_m = Ec_n.$$

For H_{nm} is a matrix, c_m and c_n are vectors and E is a constant.

To understand this "braket" notation, we treat $\langle\psi_n^{(0)}|$ as transpose vector and $|\psi_m^{(0)}\rangle$ as vector. Then we can do some algebraic manipulation to this equation:

$$\begin{aligned} H_{nm} &= \langle\psi_n^{(0)}|H_0|\psi_m^{(0)}\rangle + \langle\psi_n^{(0)}|V(x)|\psi_m^{(0)}\rangle && (4.2.4)\\ &= \langle\psi_n^{(0)}|4\left(\frac{\hbar^2\pi^2n^2}{2ma^2}\right)|\psi_m^{(0)}\rangle + \langle\psi_n^{(0)}|V(x)|\psi_m^{(0)}\rangle && (4.2.5)\\ &= \frac{2\hbar^2\pi^2n^2}{ma^2}\langle\psi_n^{(0)}|\psi_m^{(0)}\rangle + \langle\psi_n^{(0)}|V(x)|\psi_m^{(0)}\rangle && (4.2.6)\\ &= \frac{2\hbar^2\pi^2n^2}{ma^2}\delta_{nm} + H_{nm}^V. && (4.2.7) \end{aligned}$$

For $H_{nm} = \frac{1}{a}\int_0^a dxe^{(-i\frac{2\pi n}{a}x)}V(x)e^{(i\frac{2\pi n}{a}x)}$. We now can realize that, this is just the matrix version of expected value. For computer programing, we define $h_{nm} = \frac{H_{nm}}{E_{ISW}}$ (ISW stands for infinite square well). By Bloch's Theorem 2.2.2, we get

$$\psi(x+a) = e^{iKa}\psi(x).$$

For $-\pi \leqslant Ka \leqslant \pi$. Then we define $ka = 2\pi n + Ka$, therefore,

$$k = \frac{2\pi n}{a} + K.$$

The kinetic energy equals momentum squared over $2m$ and $P = \hbar k$. Therefore,

$$\begin{aligned} E &= \frac{\hbar^2k^2}{2m}\\ &= \frac{\hbar^2(\frac{2\pi n}{a} + K)^2}{2m}\\ &= \frac{\hbar^2[\frac{\pi}{n}(2n + \frac{Ka}{n})]^2}{2m}\\ &= \frac{\hbar^2\pi^2(2n + \frac{Ka}{n})^2}{2ma^2}\\ &= E_{ISW}\left(2n + \frac{Ka}{n}\right)^2. \end{aligned}$$

Now let's consider the 2D case. Here we introduce a rectangular unit cell with side lengths a_x and a_y obeying the periodicity conditions just as the 1D case.

$$\psi(x + a_x, y) = \psi(x, y),$$

$$\psi(x, y + a_y) = \psi(x, y).$$

Again, we introduce the basis to work on as:

$$\psi^{(0)}_{n_x n_y}(x, y) = \frac{1}{\sqrt{a_x a_y}} e^{\left(i\frac{2\pi n_x}{a_x}x + i\frac{2\pi n_y}{a_y}y\right)}, \tag{4.2.8}$$

where n_x, n_y are integers, with energy eigenvalues.

$$E^{(0)}_{n_x n_y}(x, y) = 4\left[n_x^2 + n_y^2\left(\frac{a_x^2}{a_y^2}\right)\right] E_{ISW} = E^{(0)}_{n_x} + E^{(0)}_{n_y}, \tag{4.2.9}$$

here we define $E_{ISW} = \frac{\hbar^2\pi^2}{2ma^2}$. Then the Hamiltonian matrix elements will be of the form:

$$\begin{aligned} H_{n_x n_y, m_x m_y} &= \langle\psi^{(0)}_{n_x n_y}|(H_0 + V(x, y))|\psi^{(0)}_{m_x m_y}\rangle & (4.2.10)\\ &= \delta_{n_x m_x}\delta_{n_y m_y} E^{(0)}_{n_x n_y} + H^V_{n_x n_y, m_x m_y}, & (4.2.11) \end{aligned}$$

where m_x and m_y are also integers, and

$$H_{n_x n_y, m_x m_y} = \langle\psi^{(0)}_{n_x n_y}|V(x, y)|\psi^{(0)}_{m_x m_y}\rangle.$$

By imposing the Bloch's condition, we get similar result as 1D case:

$$\psi(x + a_x, y) = e^{iK_x a_x}\psi(x, y),$$

$$\psi(x, y + a_y) = e^{iK_y a_y}\psi(x, y).$$

Then, we can write the total energy separately in terms of x and y:

$$E^{(0)}_{n_x} = E_{ISW}\left(2n_x + \frac{K_x a_x}{\pi}\right)^2, E^{(0)}_{n_y} = E_{ISW}\left(2n_y + \frac{K_y a_y}{\pi}\right)^2\left(\frac{a_x^2}{a_y^2}\right).$$

Then we jump into understanding the matrix mechanics and we will start with how to find eigenvalues and eigenvectors in linear algebra. To solve for the eigenvalues λ_i and the corresponding eigenvectors $\overline{x_i}$ of an $n \times n$ matrix A, do the following:

1. Multiply an $n \times n$ identity matrix by the scalar λ.
2. Subtract the identity matrix multiple from the matrix A.
3. Find the determinant of the matrix we get from last step.
4. Solve for the eigenvalues of λ that satisfy the equation: $\det(A - \lambda I) = 0$.
5. Then solve for the corresponding eigenvector to each λ.

4.3 Coding of the Hamiltonian Matrix

Here we want to use Python to compute the matrix in the 1D periodic, and let's go back to the 1D periodic potential case. We understand that H^V_{nm} is a matrix and is defined by

$$\frac{1}{a}\int_0^a dx e^{-i\frac{2\pi n}{a}x} V(x) e^{i\frac{2\pi m}{a}x}. \tag{4.3.1}$$

And we define $V(x) = V_0 \cos\left(\frac{2\pi x}{a}\right)$. Plug this back into the H^V_{nm} equation and solve the integration.

$$H^V_{nm} = \frac{1}{a}\int_0^a dx e^{-i\frac{2\pi n}{a}x} V_0 \cos\left(\frac{2\pi x}{a}\right) e^{i\frac{2\pi m}{a}x}.$$

By using Euler's identity, $\cos\left(\frac{2\pi x}{a}\right) = \frac{e^{i\frac{2\pi x}{a}} + e^{-i\frac{2\pi x}{a}}}{2}$. Hence,

$$\begin{aligned}
H^V_{nm} &= \frac{V_0}{2a}\int_0^a dx e^{-i\frac{2\pi n}{a}x}\left(e^{i\frac{2\pi x}{a}} + e^{-i\frac{2\pi x}{a}}\right) e^{i\frac{2\pi m}{a}x} \\
&= \frac{V_0}{2a}\int_0^a dx e^{-i\frac{2\pi n}{a}x} e^{i\frac{2\pi x}{a}(1+m)} + e^{i\frac{2\pi m}{a}x} e^{-i\frac{2\pi x}{a}(1+n)} \\
&= \frac{V_0}{2a}\int_0^a dx e^{i\frac{2\pi x}{a}(1+m-n)} + e^{-i\frac{2\pi x}{a}(1+n-m)} \\
&= \frac{V_0}{2a}\left[\frac{a}{i2\pi(1+m-n)} e^{i\frac{2\pi x}{a}(1+m-n)} - \frac{a}{i2\pi(1+n-m)} e^{-i\frac{2\pi x}{a}(1+n-m)}\right]_0^a \\
&= -\frac{V_0 i}{4\pi}\left[\frac{1}{(1+m-n)} e^{i\frac{2\pi x}{a}(1+m-n)} - \frac{1}{(1+n-m)} e^{-i\frac{2\pi x}{a}(1+n-m)}\right]_0^a \\
&= -\frac{V_0 i}{4\pi}\left[\frac{1}{(1+m-n)} e^{i2\pi(1+m-n)} - \frac{1}{(1+n-m)} e^{-i2\pi(1+n-m)} - \frac{1}{(1+m-n)} + \frac{1}{(1+n-m)}\right].
\end{aligned}$$

n and m are integers that indicate the rows and columns of the matrix. Therefore, $1 + m - n$ and $1 + n - m$ are both integers. Again, by using Euler's identity,

$$e^{i2\pi(1+m-n)} = e^{-i2\pi(1+n-m)} = 1.$$

Thus,

$$H^V_{nm} = -\frac{V_0 i}{4\pi}\left[\frac{1}{(1+m-n)} - \frac{1}{(1+n-m)} - \frac{1}{(1+m-n)} + \frac{1}{(1+n-m)}\right] = 0.$$

Then let's consider a special case, when $1 + m - n = 0$, which means $n = m + 1$. Therefore,

$$\begin{aligned}
H^V_{m+1m} &= \frac{1}{a}\int_0^a dx e^{-i\frac{2\pi(m+1)}{a}x} V(x) e^{i\frac{2\pi m}{a}x} \\
&= \frac{1}{a}\int_0^a dx e^{-i\frac{2\pi x}{a}} V(x) \\
&= \frac{V_0}{2a}\int_0^a dx e^{-i\frac{2\pi x}{a}} \left(e^{i\frac{2\pi x}{a}} + e^{-i\frac{2\pi x}{a}}\right) \\
&= \frac{V_0}{2a}\int_0^a dx e^{-i\frac{4\pi x}{a}} + 1 \\
&= \frac{V_0}{2a}\left(\frac{a}{-i4\pi} e^{-i\frac{4\pi x}{a}} + x\right)_0^a \\
&= \frac{V_0}{2a}\left(\frac{a}{-i4\pi} e^{-i4\pi} + a - \frac{a}{-i4\pi}\right) \\
&= \frac{V_0}{2}.
\end{aligned}$$

This calculation allows us to construct the (n, m) elements of the H^V_{nm} is a n by m matrix. The nth diagonal element equals $4\left(\frac{\hbar^2\pi^2 n^2}{2ma^2}\right)$.whereas the off diagonal elements when $n = m + 1$ are $\frac{V_0}{2}$. For the rest of the other entries in the matrix we get 0. This calculation helps us to construct the Hamiltonian matrix. Then we can input this matrix into computer and to do more manipulations.

In order to code this matrix in Python, we first define the size of the matrix and form two arrays to label the rows and the columns. And since the m's and n's are integers, they can be positive, zero or negative. But Python labels things starting from 0, so we derive a formula to convert the indexing:

$$y = \frac{1 + (-1)^{(x+1)}(2x + 1)}{4}, \tag{4.3.2}$$

where x is the python indexing and y can be either m's or n's. That is to say, we can run a forloop over over the list of m's and n's with this function and get two new lists with appropriate Python indexing. Then we construct a 0 matrix with appropriate size we defined at the beginning, and use the indexing to assign values for diagonal and off-diagonal elements that we calculated before.

Finally, we use the built-in linear algebra package “scipy.linalg” to calculate the eigenvalues and eigenvectors. For each eigenvector, we can construct a wave-function and the final step would be to plot the wave-function.

Since graphene is a 2D flat sheet, what we really want to consider is the 2-Dimensional Hamiltonian matrix. To construct the 2D Hamiltonian matrix, we need to first define the 2D wave function and the basis we are going to work on. For wave functions, we define

$$\psi(x+a_x, y) = \psi(x, y) \tag{4.3.3}$$

$$\psi(x, y+a_y) = \psi(x, y), \tag{4.3.4}$$

for similar periodicity as the 1D case. Here a_x and a_y act like a in the 1D case. Similar to the 1D case, we work in a basis like this:

$$\psi_{n_x n_y}(x, y) = \frac{1}{\sqrt{a_x a_y}} e^{\left(i\frac{2\pi n_x}{a_x}x + i\frac{2\pi n_y}{a_y}y\right)}, \tag{4.3.5}$$

where n_x, n_y are integers. From 4.2.11, we have $H_{n_x n_y, m_x m_y} = \delta_{n_x m_x}\delta_{n_y m_y}E^{(0)}_{n_x n_y} + H^V_{n_x n_y, m_x m_y}$, and we want to find $H^V_{n_x n_y, m_x m_y}$ first and then we can define the relation between n_x, n_y, m_x, m_y. With $H^V_{n_x n_y, m_x m_y} = \langle \psi^{(0)}_{n_x n_y} | V(x, y) | \psi^{(0)}_{m_x m_y}\rangle$, we have

$$H^V_{n_x n_y, m_x m_y} = \frac{1}{a_x a_y}\int_0^a \int_0^a e^{\left(i\frac{2\pi n_x}{a_x}x + i\frac{2\pi n_y}{a_y}y\right)}\ V(x, y)\ e^{\left(i\frac{2\pi m_x}{a_x}x + i\frac{2\pi m_y}{a_y}y\right)} dxdy. \tag{4.3.6}$$

We define $V(x, y)$ to be some function has periodicity: $V(x, y) = V_0\left[\cos\left(\frac{2\pi x}{a_x}\right)\cdot\cos\left(\frac{2\pi y}{a_y}\right)\right]$. By Euler's identity, we get $\cos\left(\frac{2\pi x}{a_x}\right) = \frac{e^{i\frac{2\pi x}{a_x}} + e^{-i\frac{2\pi x}{a_x}}}{2}$. Then we substitute the Euler's identity version into the expression of $V(x, y)$, and we get

$$\begin{aligned} V(x, y) &= \frac{V_0}{2}\left[\left(e^{i\frac{2\pi x}{a_x}} + e^{-i\frac{2\pi x}{a_x}}\right)\left(e^{i\frac{2\pi y}{a_y}} + e^{-i\frac{2\pi y}{a_y}}\right)\right] \\ &= \frac{V_0}{2}\left(e^{i\frac{2\pi x}{a_x}+i\frac{2\pi y}{a_y}} + e^{i\frac{2\pi x}{a_x}-i\frac{2\pi y}{a_y}} + e^{-i\frac{2\pi x}{a_x}+i\frac{2\pi y}{a_y}} + e^{-i\frac{2\pi x}{a_x}-i\frac{2\pi y}{a_y}}\right). \end{aligned}$$

Next step is to substitute this expression into the $H^V_{n_x n_y, m_x m_y}$ equation and we get

$$\begin{aligned} H^V_{n_x n_y, m_x m_y} = \frac{V_0}{2a_x a_y}\int_0^a \int_0^a e^{\left(i\frac{2\pi n_x}{a_x}x + i\frac{2\pi n_y}{a_y}y\right)}\ \left(e^{i\frac{2\pi x}{a_x}+i\frac{2\pi y}{a_y}} + e^{i\frac{2\pi x}{a_x}-i\frac{2\pi y}{a_y}} + e^{-i\frac{2\pi x}{a_x}+i\frac{2\pi y}{a_y}} + e^{-i\frac{2\pi x}{a_x}-i\frac{2\pi y}{a_y}}\right) \\ e^{\left(i\frac{2\pi m_x}{a_x}x + i\frac{2\pi m_y}{a_y}y\right)} dxdy. \end{aligned} \tag{4.3.7}$$

By combining terms, we find most of the terms are zero and left with some none zero terms which indicate the relations between n_x, m_x, n_y, m_y. We get

$$n_x + 1 + m_x = 0$$

$$n_x - 1 + m_x = 0$$

$$n_y + 1 + m_y = 0$$

$$n_y - 1 + m_y = 0.$$

We simplify these equations and find

$$n_x = \pm 1 - m_x \tag{4.3.8}$$

$$n_y = \pm 1 - m_y. \tag{4.3.9}$$

In terms of constructing the 2D Hamiltonian matrix, the trickiest thing is to find out proper indexing. Just as in the 1D case, we have to convert Python indexing. Just as the 1D case, we need to define a function to convert the indices. But now we have a pair of integers to represent another positive integer. For instance, (n_x, n_y) pair represents the rows and (m_x, m_y) pair represents the columns. How can we use these pairs to indicate one positive integer? The answer is to give the pair an order. For example, we say a positive integer always goes before a negative, so instead of using the pairs for indexing, we can use the order of the pair for indexing. For Python, we first define a function that is the square of the first element in the function plus the square of the second element. Then we insert this function in the "sort" command to make our list of pair in the order we want. Then we use the length n of the list of pairs to define the size of the Hamiltonian matrix, which is $n \times n$. Then we assign values to each entries of the matrix when the indexing of the matrix satisfies the combinations of these two relations: $n_x = \pm 1 - m_x$, $n_y = \pm 1 - m_y$. When the indexing satisfies these relations, the entries will be $\frac{V_0}{2}$. The matrix entries remain zero. Then we calculate the eigenvalues and eigenvectors. I plotted the energy levels out of the eigenvalues. Then we multiply the plane wave basis $\psi_{n_x n_y}(x, y) = \frac{1}{\sqrt{a_x a_y}} e^{\left(i \frac{2\pi n_x}{a_x} x + i \frac{2\pi n_y}{a_y} y\right)}$ with each eigenvectors. Then we sum them together and get the absolute value of the sum. Then the sum will be all real numbers and we can then make a 3D plot to see the energy eigenstates. Below are my outputs of the energy eigenstates with both imaginary and real part and the energy levels.

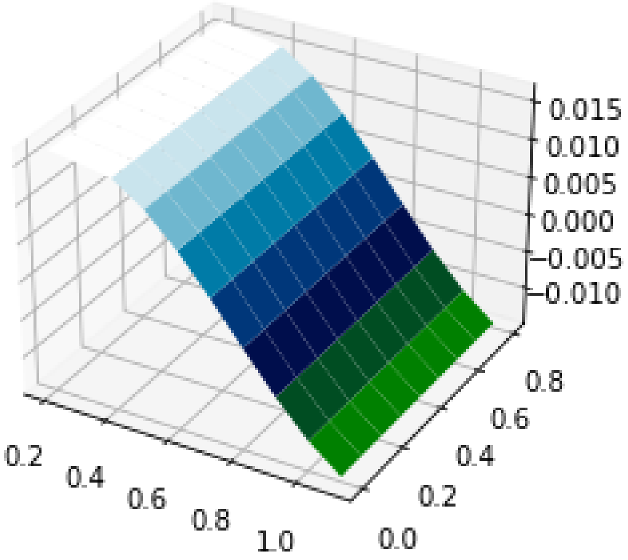


Figure 4.3.1. This is the real part of the energy eigenstate.

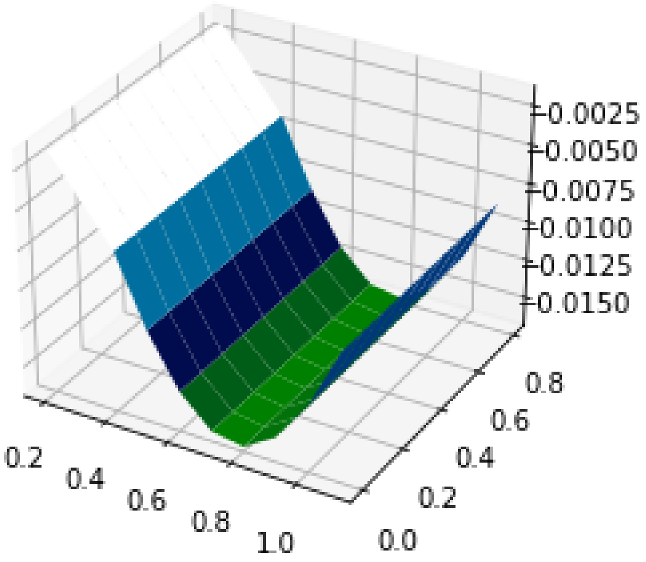


Figure 4.3.2. This is the imaginary part of the energy eigenstate.

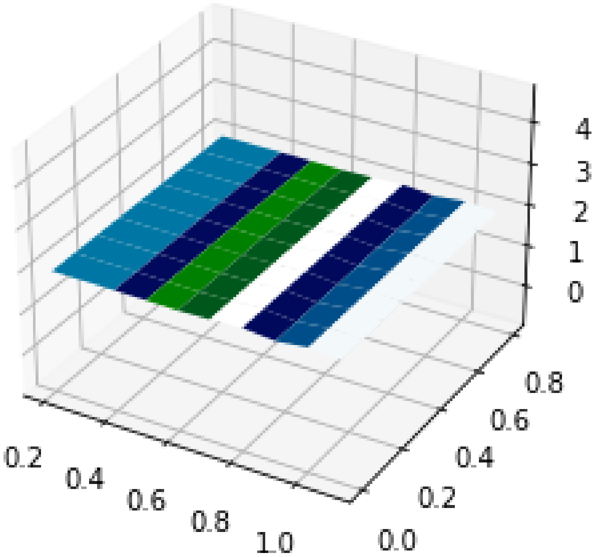


Figure 4.3.3. This is the absolute value squared of the energy eigenstates.

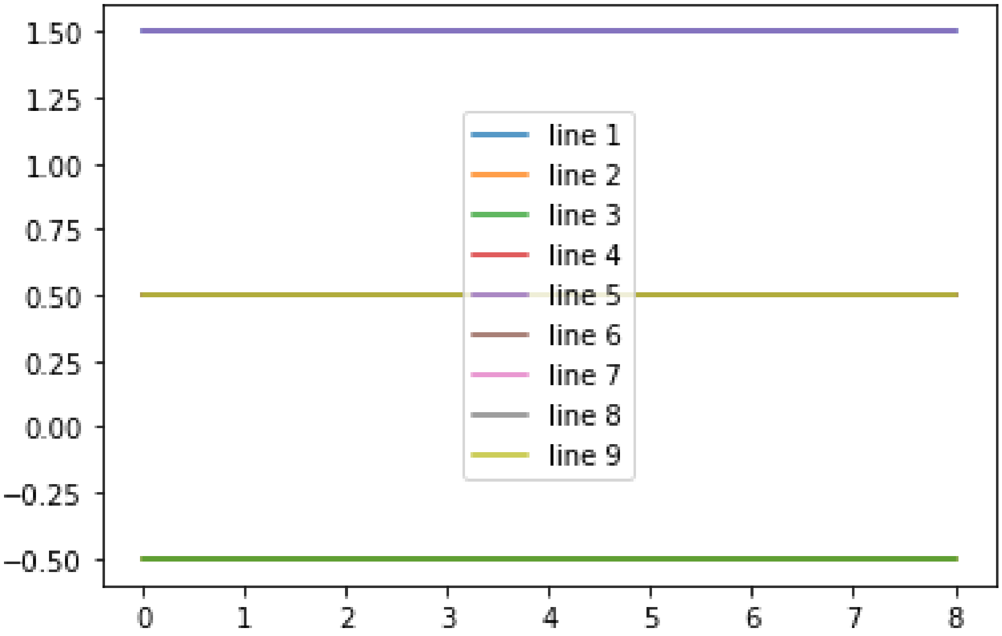


Figure 4.3.4. This figure indicates the energy levels.

5

Curvature and Future Thoughts

5.1 Gaussian Curvature

Now let me introduce curvature to this project. Intuitively, the curvature is the amount by which a curve deviates from being a straight line, or a surface deviates from being a plane. By using the osculating circle, we get the curvature(κ) equals one over the radius of the osculating circle. But what is an osculating circle? An osculating circle is the circle that nest approximate the curve at a point. Therefore the formula is:

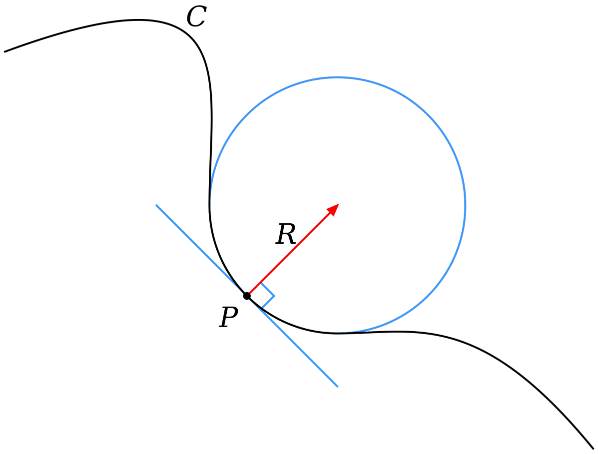


Figure 5.1.1. is from [8]. This is the osculating circle that we used to find the curvature.

$$\kappa = \frac{1}{R}. \tag{5.1.1}$$

After understanding the curvature of 1D curve, we now want to understand the Gaussian curvature of 2D shape. Since our graphene is a 2D curved sheet, what we really need to consider is the 2D curvature. The Gaussian curvature is the product of the two principal curvatures:

$$K = \kappa_1 \kappa_2. \tag{5.1.2}$$

The other formula for 2D curvature would be:

$$K = \frac{F_{xx} \cdot F_{yy} - F_{xy}^2}{(1 + F_x^2 + F_y^2)^2}. \tag{5.1.3}$$

We first find a function F that defines how the curvature changes in both x and y directions. Then $F_x = \partial x F$, $F_{xx} = \partial x \partial x F$, same for y and $F_{xy} = \partial x \partial y F$. I will introduce one example, that is to calculate the curvature of a sphere. When we define a sphere, we use $x^2 + y^2 + z^2 = R^2$, then $F(x, y) = z = \sqrt{R^2 - x^2 - y^2}$. Therefore, by our formula:

$$F_x = \partial x F = \frac{-x}{\sqrt{R^2 - x^2 - y^2}},$$

$$F_y = \partial y F = \frac{-y}{\sqrt{R^2 - x^2 - y^2}}.$$

And

$$\begin{aligned}
F_{xx} &= \partial x \partial x F \\
&= \frac{-1}{\sqrt{R^2 - x^2 - y^2}} + \left[\frac{-x \cdot (-\frac{1}{2})}{(R^2 - x^2 - y^2)^{3/2}} \cdot (-2x) \right] \\
&= \frac{-1}{\sqrt{R^2 - x^2 - y^2}} - \frac{x^2}{(R^2 - x^2 - y^2)^{3/2}}
\end{aligned}$$

$$\begin{aligned}
F_{yy} &= \partial y \partial y F \\
&= \frac{-1}{\sqrt{R^2 - x^2 - y^2}} + \left[\frac{-y \cdot (-\frac{1}{2})}{(R^2 - x^2 - y^2)^{3/2}} \cdot (-2y) \right] \\
&= \frac{-1}{\sqrt{R^2 - x^2 - y^2}} - \frac{y^2}{(R^2 - x^2 - y^2)^{3/2}}
\end{aligned}$$

$$\begin{aligned}
F_{xy} &= \partial x \partial y F \\
&= \partial x \left(\frac{-y}{\sqrt{R^2 - x^2 - y^2}} \right) \\
&= \frac{-y \cdot (-\frac{1}{2})}{(R^2 - x^2 - y^2)^{3/2}} \cdot (-2x) = \frac{-xy}{(R^2 - x^2 - y^2)^{3/2}}.
\end{aligned}$$

Then we plug those items into the curvature formula and we get:

$$\begin{aligned}
K &= \frac{F_{xx}\cdot F_{yy} - F_{xy}^2}{(1+F_x^2+F_y^2)^2}\\
&= \left(\frac{-1}{\sqrt{R^2-x^2-y^2}} - \frac{x^2}{(R^2-x^2-y^2)^{3/2}}\right)\cdot\left(\frac{-1}{\sqrt{R^2-x^2-y^2}} - \frac{y^2}{(R^2-x^2-y^2)^{3/2}}\right)\\
&\qquad - \left(\frac{-xy}{(R^2-x^2-y^2)^{3/2}}\right)^2 \div \left[1+\left(\frac{-x}{\sqrt{R^2-x^2-y^2}}\right)^2+\left(\frac{-y}{\sqrt{R^2-x^2-y^2}}\right)^2\right]^2.
\end{aligned}$$

We calculate the numerator first:

$$\begin{aligned}
F_{xx}\cdot F_{yy} - F_{xy}^2 &= \frac{1}{R^2-x^2-y^2} + \frac{y^2}{(R^2-x^2-y^2)^2} + \frac{x^2}{(R^2-x^2-y^2)^2} + \frac{x^2y^2}{(R^2-x^2-y^2)^3} - \frac{x^2y^2}{(R^2-x^2-y^2)^3}\\
&= \frac{R^2-x^2-y^2+y^2+x^2}{(R^2-x^2-y^2)^2}\\
&= \frac{R^2}{(R^2-x^2-y^2)^2}.
\end{aligned}$$

Then we calculate the denominator:

$$\begin{aligned}
\frac{1}{(1+F_x^2+F_y^2)^2} &= \frac{1}{\left[1+\left(\frac{-x}{\sqrt{R^2-x^2-y^2}}\right)^2+\left(\frac{-y}{\sqrt{R^2-x^2-y^2}}\right)^2\right]^2}\\
&= \frac{1}{\left(1+\frac{x^2+y^2}{(R^2-x^2-y^2)}\right)^2}\\
&= \frac{1}{1+\frac{2x^2+2y^2}{(R^2-x^2-y^2)}+\frac{(x^2+y^2)^2}{(R^2-x^2-y^2)^2}}\\
&= \frac{(R^2-x^2-y^2)^2}{R^4}.
\end{aligned}$$

Then we put the numerator and denominator together, we get:

$$K = \frac{R^2}{(R^2-x^2-y^2)^2}\cdot\frac{(R^2-x^2-y^2)^2}{R^4} = \frac{1}{R^2}. \tag{5.1.4}$$

By definition of Gaussian curvature, $K_{sphere} = \kappa_1\kappa_2 = \frac{1}{R}\frac{1}{R} = \frac{1}{R^2}$, which agrees what we found by parametrizing the sphere.

Another example would be to calculate the curvature of a cylinder.

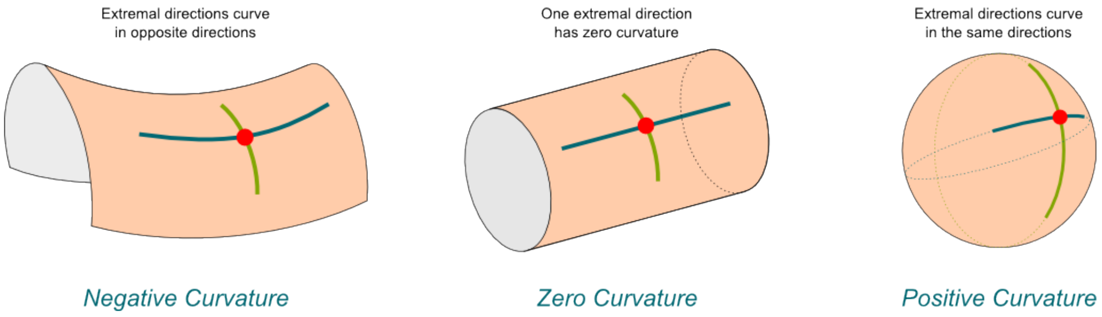


Figure 5.1.2. is from [12]. In the middle of the figure, we can tell the curvature from the horizontal direction of the cylinder is zero.

From 5.1.2, we get $K_{\text{cylinder}} = \kappa_1\kappa_2$ and from Fig 5.1.2, we know $\kappa_2 = 0$. Therefore the Gaussian curvature of the cylinder is 0.

5.2 Conclusion and Future Thoughts

In conclusion, this project has set up the strategy towards how to explore the electronic properties of curved graphene sheet. This project begins with set up the basic knowledge of solid state physics. We introduced crystal structure with fine discussions on Bravais lattice, reciprocal lattice and Brillouin zone. These help us build the mind set of treating lattice as vectors, thinking lattice with periodicity and due to periodicity we can constraint the momentum and potential of the whole lattice within the first Brillouin zone and this will save us a lot of time. Next, we introduced what is band structure and the use of it, then we specifically mentioned the unique band structure graphene has, which including introduce the Dirac cone. Then we proof an important theorem used throughout this project, the Bloch's theorem, which tells us:

$$\psi_n(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}\psi_n(\mathbf{r}),$$

and helps us to explore the periodicity within the lattice. The next step is to understand the potentials of the flat sheet graphene. We introduced the perturbation theory which will help us find the potential when the wave function is tilted a little bit and this theory sets up a strong base for us to understand the potential models introduced later. Then we came to two potential models: the weak potential and the tight-binding model and the tight-binding model is more suitable for graphene due to its consideration of insulator. Hence, this project further discussed the application of tight-binding model on graphene and we successfully find the energy levels of flat sheet graphene.

Next, I constructed the Hamiltonian matrix in 1D and 2D with periodic potential $\cos(x)$. I used Python to find the eigenvalues and eigenvectors of the Hamiltonian matrix and then plot the energy levels. Then I multiply the eigenvectors with the plane wave basis and sum them to plot the specific states. I also wrote another code that will do the numerical integration with different periodic potentials. The next step would be to diagonalize the matrix with different potential and repeat the process of plotting the energy levels and the state. Then I briefly introduced the the Gaussian curvature and examine the curvature of the sphere and the cylinder.

Future thoughts diverge in two directions: first is to consider the "buckyball" structure; the second is to consider the nano tube. For the buckyball structure, since its a ball we know the gaussian curvature of it but the periodicity of the potential is more complicated, because it has mixture of pentagons and hexagons. For the nano tube, it is a cylinder, so it has no gaussian curvature, so we have to change the boundary conditions to parametrize the curved side. For example, we need to define a function between x and y to tell how the boundary changes. Below attached two figures describing the shapes of buckyball and nano tube.

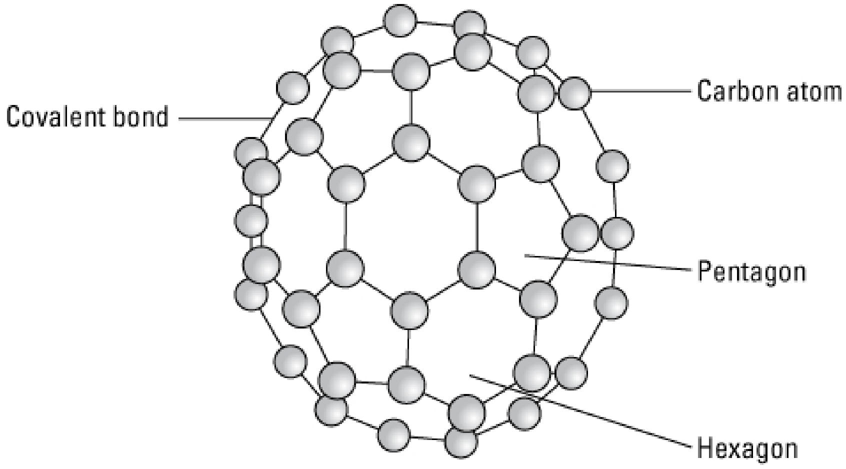


Figure 5.2.1. is from [6]. This is the buckyball shape.

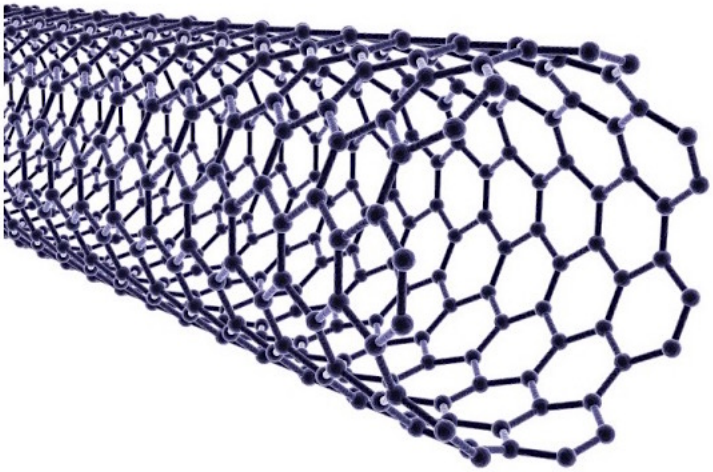


Figure 5.2.2. is from [7]. This is the nanotube.

Then we would first find the periodic potential of these two shapes and then construct the Hamiltonian matrix. Then we upload this new matrix to our code and it will automatically generate the energy levels and the state. So the key thing next is to figure out how the boundary changes and how the curvature affects the symmetry and periodicity.

In terms of coding, the future steps will be to try more different potentials with the integration code I wrote. Then I will get new Hamiltonian matrices. With a more general potential, I can diagonalize the Hamiltonian matrix and find the eigenvalues and eigenvectors using the linear algebra package. Then I can upload the eigenvalues and eigenvectors to my old code and it will automatically generated plots for eigenstates and energy levels. I would also like to make my code for plotting the energy levels more efficient, that is to loop through the list of all eigenvalues and make a single plot. So no matter what is the size of the eigenvalue list, I can always plot the energy levels as a single graph.

At the end, if future physics students are interested in graphene, I hope this project is helpful to you. And please consider at least 10 times before you decide to continue doing this project. It is going to be a really hard project, but with many joy when you are learning new things about condensed-matter physics. Good Luck!

Appendix A

Python Code

1D Hamiltonian Matrix

May 4, 2021

[44]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
import scipy.linalg as la
```

[76]:
```
P=12  #Here is to define the size of our matrix
```

[77]:
```
row=np.arange(P)#Define how many rows the Hamiltonian matrix have
column=np.arange(P)#Define how many columms the Hamiltonian matrix have
```

[78]:
```
def x1(a):
    y1=4*a**2
    return y1 #here is to calculate the diagonal value: multiples of the energy␣
 ↪of infinite square well
```

[79]:
```
def n(x):
    y=(1+(-1)**(x+1)*(2*x+1))/4 #Here is to define a function that allow us to␣
 ↪use all
    return y                    #positive integers we defined before to␣
 ↪represent the negative indexing
                                #We allow the Hamiltonian matrix to have␣
 ↪negative indexing.
```

[80]:
```
def V0(m):
    v=m/2
    return v#Here is to define when we off the diagonal by 1, what values␣
 ↪should we get after
            #dividing the energy by the energy of infinite square well.
```

[81]:
```
H=np.zeros((P, P))
print (H)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

[82]:

```python
for a in row:
    for b in column:
        if n(a)==n(b):       #By plugging in function n(x), we now convert the␣
↪positive indexing to negative indexing
            H[a][b]=x1(n(a)) # Here I present the diagonal values.
        else:
            H[a][b]=0
            if n(a)==n(b)+1:
                H[a][b]=V0(1)
            elif n(a)==n(b)-1: #Here I present the off diagonal values.
                H[a][b]=V0(1)
            else:
                H[a][b]=0
print (H)
```

```
[[  0.    0.5   0.5   0.    0.    0.    0.    0.    0.    0.    0.    0. ]
 [  0.5   4.    0.    0.5   0.    0.    0.    0.    0.    0.    0.    0. ]
 [  0.5   0.    4.    0.    0.5   0.    0.    0.    0.    0.    0.    0. ]
 [  0.    0.5   0.   16.    0.    0.5   0.    0.    0.    0.    0.    0. ]
 [  0.    0.    0.5   0.   16.    0.    0.5   0.    0.    0.    0.    0. ]
 [  0.    0.    0.    0.5   0.   36.    0.    0.5   0.    0.    0.    0. ]
 [  0.    0.    0.    0.    0.5   0.   36.    0.    0.5   0.    0.    0. ]
 [  0.    0.    0.    0.    0.    0.5   0.   64.    0.    0.5   0.    0. ]
 [  0.    0.    0.    0.    0.    0.    0.5   0.   64.    0.    0.5   0. ]
 [  0.    0.    0.    0.    0.    0.    0.    0.5   0.  100.    0.    0.5]
 [  0.    0.    0.    0.    0.    0.    0.    0.    0.5   0.  100.    0. ]
 [  0.    0.    0.    0.    0.    0.    0.    0.    0.    0.5   0.  144. ]]
```

By observing the Hamiltonian matrix, we find it satisfys with the property that the "Hamiltonian operator is hermitian". That is if we take the transpose conjugate of the matrix, we still get itself. And from the 3x3 case, it does satisfy this property, and it's eigenvalues should be all real.

[83]:

```python
evals, evecs = la.eig(H)
print(evals)
```

```
[-1.21765545e-01+0.j  1.44005681e+02+0.j  4.10090060e+00+0.j
  1.00006944e+02+0.j  1.00001263e+02+0.j  6.40019842e+01+0.j
  6.40019848e+01+0.j  3.60035717e+01+0.j  3.60035717e+01+0.j
  3.97918922e+00+0.j  1.60083646e+01+0.j  1.60083105e+01+0.j]
```

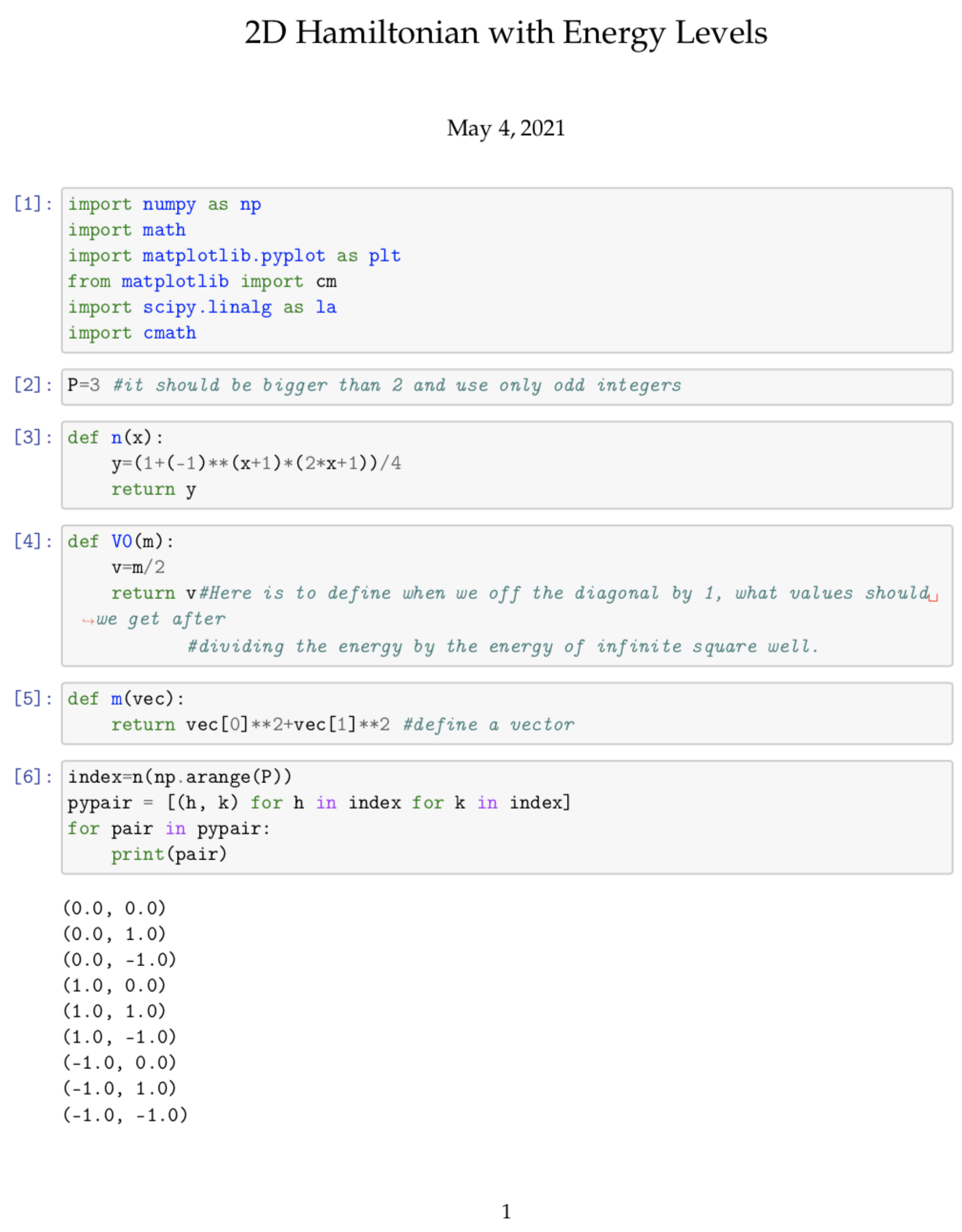
Here we see that all the eigenvalues are real and hence we did a great job in constructing the 1D

Hamiltonian matrix.

```
[84]: print(evecs)
```

```
[[ 9.85481299e-01 -1.59780127e-14  1.69774020e-01  1.68132238e-11
  -1.68193732e-11 -1.21097970e-08 -1.21097973e-08 -7.67188518e-06
  -8.15498840e-08 -1.34236783e-15  1.84179277e-03  7.76149329e-14]
 [-1.19997667e-01 -4.58620640e-12  6.96226380e-01  8.77723278e-14
  -3.36382336e-09 -1.55000119e-06 -1.00964480e-10 -2.73280214e-04
  -2.79031451e-04 -7.06494947e-01  2.94840902e-02  2.94077174e-02]
 [-1.19997667e-01 -4.15007948e-16  6.96226380e-01  3.36278471e-09
  -8.77442348e-14 -1.00876334e-10 -1.55000115e-06 -2.79150322e-04
   2.73159277e-04  7.06494947e-01  2.94840902e-02 -2.94077174e-02]
 [ 3.72320294e-03 -1.28411486e-09 -2.92747073e-02  7.34451306e-16
  -6.45845704e-07 -1.85994184e-04 -6.33430205e-12 -1.74842140e-02
  -1.78599246e-02  2.94054278e-02  7.06269619e-01  7.06274002e-01]
 [ 3.72320294e-03 -2.21091649e-16 -2.92747073e-02  6.45684506e-07
  -7.55408967e-16  4.23097419e-12 -1.85994181e-04 -1.78599428e-02
   1.74842266e-02 -2.94054278e-02  7.06269619e-01 -7.06274002e-01]
 [-5.15423985e-05 -3.28743408e-07  4.58924214e-04 -8.19939562e-16
  -1.08500346e-04 -1.78546297e-02 -5.07139710e-10 -6.99220175e-01
  -7.14245531e-01 -4.59220938e-04 -1.76687324e-02 -1.76687942e-02]
 [-5.15423985e-05  4.41419007e-16  4.58924214e-04  1.08480601e-04
  -1.06605629e-15  5.07103894e-10 -1.78546297e-02 -7.14246143e-01
   6.99220800e-01  4.59220938e-04 -1.76687324e-02  1.76687942e-02]
 [ 4.01925970e-07 -7.10110277e-05 -3.83097737e-06 -9.33481682e-14
  -1.38876725e-02 -9.99744124e-01 -2.83955471e-08  1.24894103e-02
   1.27577919e-02  3.82568024e-06  1.84092792e-04  1.84093228e-04]
 [ 4.01925970e-07  4.18725139e-17 -3.83097737e-06  1.38863778e-02
  -9.18885482e-14  2.83955480e-08 -9.99744144e-01  1.27578027e-02
  -1.24894214e-02 -3.82568023e-06  1.84092792e-04 -1.84093228e-04]
 [-2.00722057e-09 -1.13622426e-02  1.99743729e-08 -6.66522149e-12
  -9.99838998e-01  1.38873060e-02  3.94437975e-10 -9.75824930e-05
  -9.96794173e-05 -1.99214698e-08 -1.09592504e-06 -1.09592693e-06]
 [-2.00718579e-09  4.52640989e-16  1.99740007e-08  9.99903574e-01
  -6.66342522e-12 -3.94404744e-10  1.38861009e-02 -9.96758965e-05
   9.75790500e-05  1.99210995e-08 -1.09589956e-06  1.09590144e-06]
 [ 6.96363524e-12 -9.99935445e-01 -7.13886914e-11  7.58381590e-14
   1.13621329e-02 -8.67978152e-05 -2.46508319e-12  4.51785742e-07
   4.61494046e-07  7.11375570e-11  4.28123689e-09  4.28124257e-09]]
```

```
[ ]:
```



2D Hamiltonian with Energy Levels

May 4, 2021

[1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib import cm
import scipy.linalg as la
import cmath
```

[2]:
```
P=3 #it should be bigger than 2 and use only odd integers
```

[3]:
```
def n(x):
    y=(1+(-1)**(x+1)*(2*x+1))/4
    return y
```

[4]:
```
def V0(m):
    v=m/2
    return v#Here is to define when we off the diagonal by 1, what values should␣
 ↪we get after
            #dividing the energy by the energy of infinite square well.
```

[5]:
```
def m(vec):
    return vec[0]**2+vec[1]**2 #define a vector
```

[6]:
```
index=n(np.arange(P))
pypair = [(h, k) for h in index for k in index]
for pair in pypair:
    print(pair)
```

```
(0.0, 0.0)
(0.0, 1.0)
(0.0, -1.0)
(1.0, 0.0)
(1.0, 1.0)
(1.0, -1.0)
(-1.0, 0.0)
(-1.0, 1.0)
(-1.0, -1.0)
```

[7]:
```
pypair=sorted (pypair, key=m)
```

[8]:
```
print(pypair)
```

```
[(0.0, 0.0), (0.0, 1.0), (0.0, -1.0), (1.0, 0.0), (-1.0, 0.0), (1.0, 1.0), (1.0,
-1.0), (-1.0, 1.0), (-1.0, -1.0)]
```

[9]:
```
H_2D=np.zeros((len(pypair), len(pypair)))
```

[11]:
```
size=np.arange(len(pypair)) #we use the length of the sorted pair to label the␣
 ↪entries of the matrix
```

[13]:
```
for x in size:
    for y in size:
        if pypair[x][0]==1-pypair[y][0] and pypair[x][1]==1-pypair[y][1]:
            H_2D[x][y]=V0(1)
        elif pypair[x][0]==-1-pypair[y][0] and pypair[x][1]==-1-pypair[y][1]:
            H_2D[x][y]=V0(1)
        elif pypair[x][0]==1-pypair[y][0] and pypair[x][1]==-1-pypair[y][1]:
            H_2D[x][y]=V0(1)
        elif pypair[x][0]==-1-pypair[y][0] and pypair[x][1]==1-pypair[y][1]:
            H_2D[x][y]=V0(1)
        elif pypair[x][0]==pypair[y][0] and pypair[x][1]==pypair[y][1]:
            H_2D[x][y]=V0(1)
        else:
            H_2D[x][y]=0
print (H_2D)
```

```
[[0.5 0.  0.  0.  0.  0.5 0.5 0.5 0.5]
 [0.  0.5 0.  0.5 0.5 0.  0.  0.  0. ]
 [0.  0.  0.5 0.5 0.5 0.  0.  0.  0. ]
 [0.  0.5 0.5 0.5 0.  0.  0.  0.  0. ]
 [0.  0.5 0.5 0.  0.5 0.  0.  0.  0. ]
 [0.5 0.  0.  0.  0.  0.5 0.  0.  0. ]
 [0.5 0.  0.  0.  0.  0.  0.5 0.  0. ]
 [0.5 0.  0.  0.  0.  0.  0.  0.5 0. ]
 [0.5 0.  0.  0.  0.  0.  0.  0.  0.5]]
```

[14]:
```
evals, evecs = la.eig(H_2D)
print(evals)
```

```
[ 1.5+0.j -0.5+0.j -0.5+0.j  0.5+0.j  1.5+0.j  0.5+0.j  0.5+0.j  0.5+0.j
  0.5+0.j]
```

[15]:
```
print(evecs)
```

```
[[ 7.07106781e-01  7.07106781e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

```
  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  5.00000000e-01 -7.07106781e-01
   5.00000000e-01  2.98519562e-01 -1.78460876e-01  6.51380093e-02
   1.43106624e-01]
 [ 0.00000000e+00  0.00000000e+00  5.00000000e-01  7.07106781e-01
   5.00000000e-01 -2.98519562e-01  1.78460876e-01 -6.51380093e-02
  -1.43106624e-01]
 [ 0.00000000e+00  0.00000000e+00 -5.00000000e-01  1.73014897e-16
   5.00000000e-01  6.41003956e-01 -2.34821407e-17 -5.76024054e-02
   2.24449320e-01]
 [ 0.00000000e+00  0.00000000e+00 -5.00000000e-01 -7.24202417e-17
   5.00000000e-01 -6.41003956e-01  9.24412538e-18  5.76024054e-02
  -2.24449320e-01]
 [ 3.53553391e-01 -3.53553391e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  6.51770155e-01  5.85132062e-01
   6.54500022e-02]
 [ 3.53553391e-01 -3.53553391e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  3.10366740e-02 -3.13569447e-01
  -7.44721884e-01]
 [ 3.53553391e-01 -3.53553391e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  3.10366740e-02 -6.39417858e-01
   5.24955209e-01]
 [ 3.53553391e-01 -3.53553391e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00 -7.13843503e-01  3.67855243e-01
   1.54316673e-01]]
```

[16]:

```
a_x=1
```

[17]:

```
a_y=2
```

[29]:

```
def Psi(x,y,n_x,n_y):
    S=1/(np.sqrt(a_x)*np.sqrt(a_y))*np.exp(2*np.pi*n_x*x*1j/a_x+2*np.pi*n_y*y*1j/
↪a_y)
    return S
```

[30]:

```
xvals=[0.2,0.4,0.5,0.6,0.7,0.8,0.9,1,1.1]
yvals=[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]
```

[31]:

```
array=(np.zeros((9,9),dtype=np.complex_),) * 9
```

[32]:

```
for m in size:
    for i in size:
        for j in size:
            array[m][i][j]=Psi(xvals[i],yvals[j],pypair[m][0],pypair[m][1])
```

[33]:

```
print (np.dot(evecs[2],array))
```

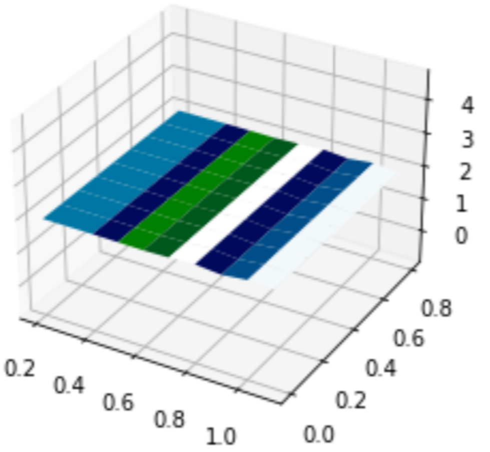
```
[[-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]
 [-0.95837954+0.56304004j -0.73748416+0.83163846j -0.4443987 +1.01883031j
  -0.1078124 +1.10629196j  0.23932733+1.08546204j  0.56304004+0.95837954j
   0.83163846+0.73748416j  1.01883031+0.4443987j   1.10629196+0.1078124j ]]
```

```
[34]: print (np.abs(np.dot(evecs[2],array)))
```

```
[[1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]
 [1.11153291 1.11153291 1.11153291 1.11153291 1.11153291 1.11153291
  1.11153291 1.11153291 1.11153291]]
```

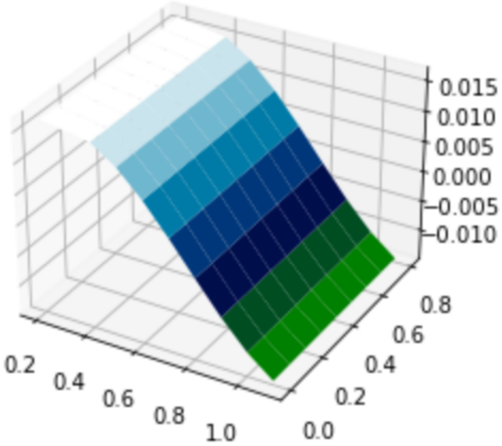
[35]:

```
fig = plt.figure()
ax = fig.gca(projection='3d')

x = xvals
y = yvals
x, y = np.meshgrid(x, y)
L= np.abs(np.dot(evecs[4],array))**2
ax.plot_surface(x, y, L, cmap=cm.ocean)
plt.show()
```



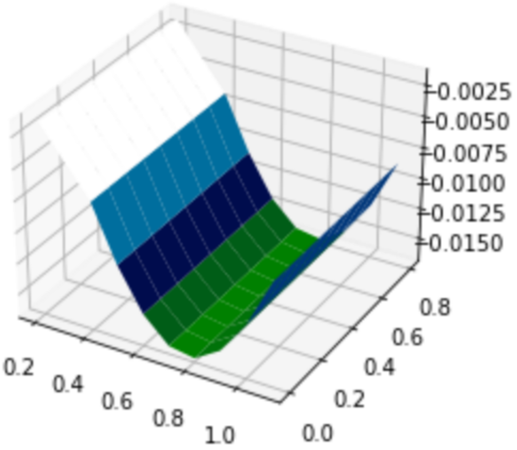
[36]:

```
fig = plt.figure()
ax = fig.gca(projection='3d')

x = xvals
y = yvals
x, y = np.meshgrid(x, y)
L= np.dot(evecs[4],array).real
ax.plot_surface(x, y, L, cmap=cm.ocean)
plt.show()
```

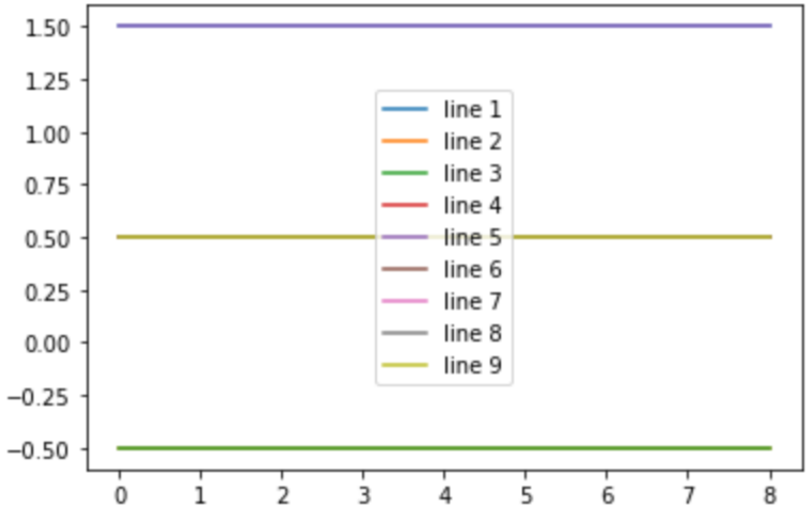


[37]:

```
fig = plt.figure()
ax = fig.gca(projection='3d')

x = xvals
y = yvals
x, y = np.meshgrid(x, y)
L= np.dot(evecs[4],array).imag
ax.plot_surface(x, y, L, cmap=cm.ocean)
plt.show()
```



[28]:
```
xvals=size
yvals=[0]*9
plt.plot(xvals, yvals+np.real(evals[0]), label = "line 1")
plt.plot(xvals, yvals+np.real(evals[1]), label = "line 2")
plt.plot(xvals, yvals+np.real(evals[2]), label = "line 3")
plt.plot(xvals, yvals+np.real(evals[3]), label = "line 4")
plt.plot(xvals, yvals+np.real(evals[4]), label = "line 5")
plt.plot(xvals, yvals+np.real(evals[5]), label = "line 6")
plt.plot(xvals, yvals+np.real(evals[6]), label = "line 7")
plt.plot(xvals, yvals+np.real(evals[7]), label = "line 8")
plt.plot(xvals, yvals+np.real(evals[8]), label = "line 9")
plt.legend()
plt.show()
```


1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
1
2
3
4
5
6
7
8
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9

```
[ ]:
```

Potential Integration

May 4, 2021

[1]:
```
import scipy.integrate as integrate
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
P=9
```

[3]:
```
def Psi(x,y,n_x,n_y):
    S=1/(np.sqrt(a_x)*np.sqrt(a_y))*(x+y)*np.exp(2*np.pi*n_x*x*1j/a_x+2*np.
 ↪pi*n_x*y*1j/a_y)
    return S
```

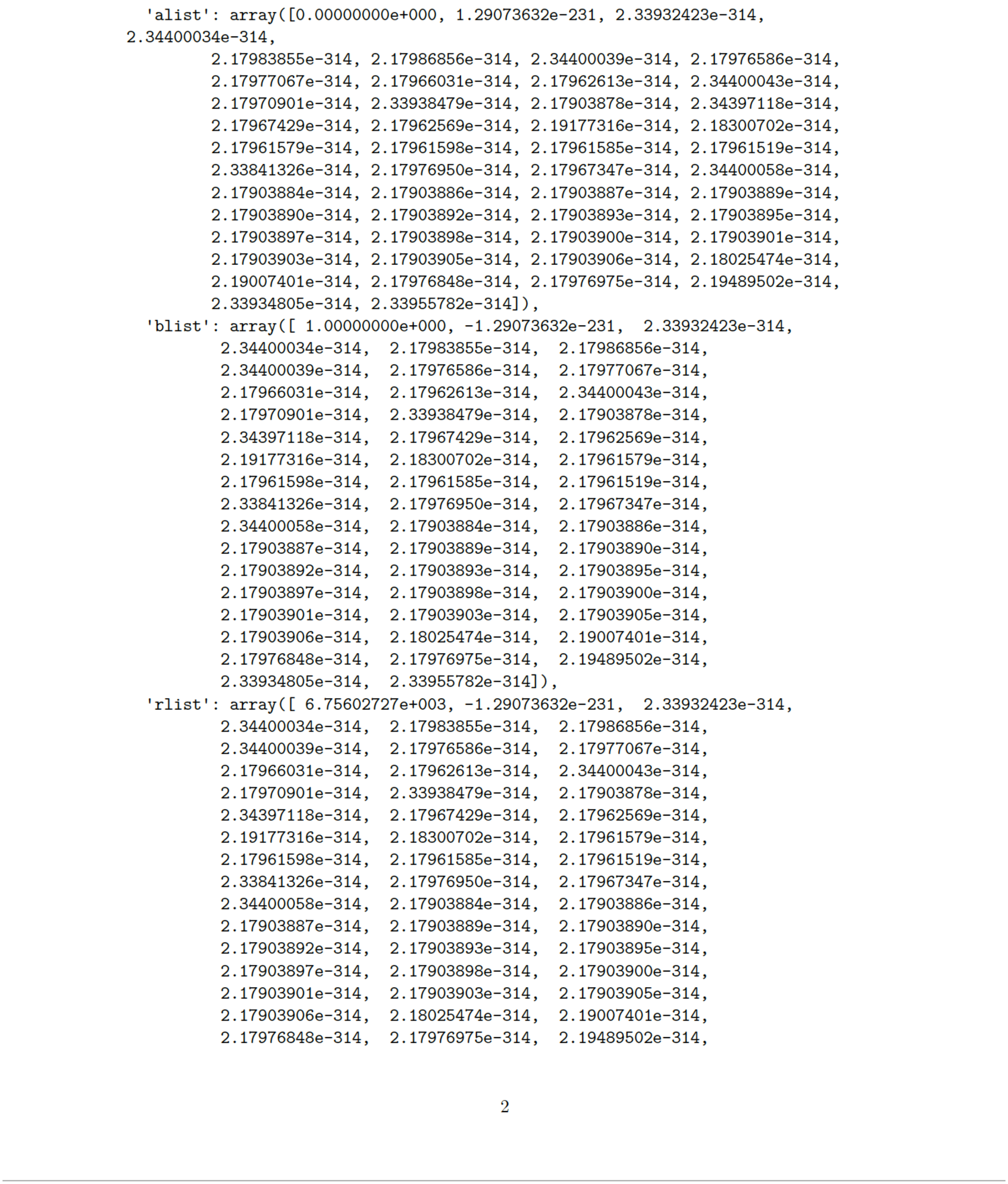
[4]:
```
a_x=1
```

[5]:
```
a_y=2
```

[8]:
```
xvals=np.arange(P)
yvals=np.arange(P)
H = [Psi(x,y,1,2)*np.cos(x)*(x+y)*Psi(x,y,3,4) for x in xvals for y in yvals]
```

[9]:
```
integrate.quad(lambda x,y: abs(sum(H)),0,1,0,1)
```

[9]:
```
(6756.0272693089,
 7.500697029382164e-11,
 {'neval': 21,
  'last': 1,
  'iord': array([          1,  1610612736,           0,  1610612736,
450428942,
                   1,   450467008,           1,           0,  1610612736,
          1103803375, -1610610696,   439549964,           1,   446892400,
                   1,   446905440,           1,   439949632,           1,
           439955568,           1,   443064432,           1,   449289456,
                   1,   449629200,           1,   439879408,           1,
           439949472,           1,   439958064,           1,   136498288,
                   1,   439942832,           1,   439957040,           1,
           433562224,           1,   449607920,           1,   439673648,
                   1,   430134128,           1,   448442160,           1],
        dtype=int32),
```



```
  'alist': array([0.00000000e+000, 1.29073632e-231, 2.33932423e-314,
2.34400034e-314,
         2.17983855e-314, 2.17986856e-314, 2.34400039e-314, 2.17976586e-314,
         2.17977067e-314, 2.17966031e-314, 2.17962613e-314, 2.34400043e-314,
         2.17970901e-314, 2.33938479e-314, 2.17903878e-314, 2.34397118e-314,
         2.17967429e-314, 2.17962569e-314, 2.19177316e-314, 2.18300702e-314,
         2.17961579e-314, 2.17961598e-314, 2.17961585e-314, 2.17961519e-314,
         2.33841326e-314, 2.17976950e-314, 2.17967347e-314, 2.34400058e-314,
         2.17903884e-314, 2.17903886e-314, 2.17903887e-314, 2.17903889e-314,
         2.17903890e-314, 2.17903892e-314, 2.17903893e-314, 2.17903895e-314,
         2.17903897e-314, 2.17903898e-314, 2.17903900e-314, 2.17903901e-314,
         2.17903903e-314, 2.17903905e-314, 2.17903906e-314, 2.18025474e-314,
         2.19007401e-314, 2.17976848e-314, 2.17976975e-314, 2.19489502e-314,
         2.33934805e-314, 2.33955782e-314]),
  'blist': array([ 1.00000000e+000, -1.29073632e-231,  2.33932423e-314,
          2.34400034e-314,  2.17983855e-314,  2.17986856e-314,
          2.34400039e-314,  2.17976586e-314,  2.17977067e-314,
          2.17966031e-314,  2.17962613e-314,  2.34400043e-314,
          2.17970901e-314,  2.33938479e-314,  2.17903878e-314,
          2.34397118e-314,  2.17967429e-314,  2.17962569e-314,
          2.19177316e-314,  2.18300702e-314,  2.17961579e-314,
          2.17961598e-314,  2.17961585e-314,  2.17961519e-314,
          2.33841326e-314,  2.17976950e-314,  2.17967347e-314,
          2.34400058e-314,  2.17903884e-314,  2.17903886e-314,
          2.17903887e-314,  2.17903889e-314,  2.17903890e-314,
          2.17903892e-314,  2.17903893e-314,  2.17903895e-314,
          2.17903897e-314,  2.17903898e-314,  2.17903900e-314,
          2.17903901e-314,  2.17903903e-314,  2.17903905e-314,
          2.17903906e-314,  2.18025474e-314,  2.19007401e-314,
          2.17976848e-314,  2.17976975e-314,  2.19489502e-314,
          2.33934805e-314,  2.33955782e-314]),
  'rlist': array([ 6.75602727e+003, -1.29073632e-231,  2.33932423e-314,
          2.34400034e-314,  2.17983855e-314,  2.17986856e-314,
          2.34400039e-314,  2.17976586e-314,  2.17977067e-314,
          2.17966031e-314,  2.17962613e-314,  2.34400043e-314,
          2.17970901e-314,  2.33938479e-314,  2.17903878e-314,
          2.34397118e-314,  2.17967429e-314,  2.17962569e-314,
          2.19177316e-314,  2.18300702e-314,  2.17961579e-314,
          2.17961598e-314,  2.17961585e-314,  2.17961519e-314,
          2.33841326e-314,  2.17976950e-314,  2.17967347e-314,
          2.34400058e-314,  2.17903884e-314,  2.17903886e-314,
          2.17903887e-314,  2.17903889e-314,  2.17903890e-314,
          2.17903892e-314,  2.17903893e-314,  2.17903895e-314,
          2.17903897e-314,  2.17903898e-314,  2.17903900e-314,
          2.17903901e-314,  2.17903903e-314,  2.17903905e-314,
          2.17903906e-314,  2.18025474e-314,  2.19007401e-314,
          2.17976848e-314,  2.17976975e-314,  2.19489502e-314,
```

```
          2.33934805e-314,  2.33955782e-314]),
  'elist': array([7.50069703e-011, 1.73059446e-077, 2.19824855e-314,
0.00000000e+000,
         6.92706684e-310, 0.00000000e+000, 0.00000000e+000, 4.94065646e-324,
         0.00000000e+000, 2.12199580e-314, 0.00000000e+000,             nan,
         2.58883486e-312, 6.36598746e-314, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
         0.00000000e+000, 2.17896818e-314, 2.19872338e-314, 2.17903893e-314,
         0.00000000e+000, 0.00000000e+000, 2.17896818e-314, 2.34485706e-314,
         2.17894183e-314, 2.17892957e-314])})
```

Bibliography

[1] Neil W. Ashcroft, *Solid State Physics*, Brooks/Cole, Cengage Learning, Belmont, 1976.

[2] David J. Griffiths, *Introduction to Quantum Mechanics*, Cambridge University Press, University Printing House, Cambridge, United Kingdom, 2018.

[3] Randy Harris, *Modern Physics*, Pearson Education, 1031 Sansome St., San Fransisco, CA, 2008.

[4] Charles Kittel, *Introduction to Solid State Physics*, John Wiley Sons, Inc, United States, 1996.

[5] M. O. Goerbig, *Electronic properties of graphene in a strong magnetic field*, Reviews of Modern Physics **83** (2011), 1193–1243.

[6] *What is a buckyball*, `https://www.understandingnano.com/what-is-buckyball-c60.html`.

[7] Stefanie Harvey, *Promising Research for Carbon Nanotube Technology in Electrical Cabling Fabrication*, `https://www.allaboutcircuits.com/industry-articles/carbon-nanotube-technology-promises-a-revolution-in-cabling/`.

[8] *Osculating Circle*, `https://en.wikipedia.org/wiki/Osculating_circle`.

[9] *Wave Number in Finite Square Well*, `https://line.17qq.com/articles/suehqcaahx_p2.html`.

[10] *Band Structure Graphene*, `http://phelafel.technion.ac.il/~tzipora/band_theory.html`.

[11] F. Guinea A. H. Castro Neto N. M. R. Peres, *The electronic properties of graphene.* . https://journals.aps.org/rmp/abstract/10.103/RevModPhys.

[12] Brian Keng, *Hyperbolic Geometry and Poincaré Embeddings*, `https://bjlkeng.github.io/posts/hyperbolic-geometry-and-poincare-embeddings/`.

[13] *Semiconductors/What is a Semiconductor*, `https://en.wikibooks.org/wiki/Semiconductors/What_is_a_Semiconductor`.