The Landau Problem

Anurag Pallaprolu
PHYS1211-I
Summer Research Fellow
Indian Institute of Science, Bengaluru - 560012

July 24, 2015

Contents

1 Acknowledgements

I would like to wholeheartedly thank the Indian Academy of Sciences for providing me with this opportunity to work at the cutting edge theoretical research being done in Physics under the Joint Summer Research Fellowship programme. I would like to thank Professor Vijay Shenoy for guiding me through a remarkably new way of doing physics by actually configuring and "re-discovering" the physical situation by myself. I would also thank Mr. Adhip Agarwala, Research Student, CCMT, IISc, for helping me out whenever I couldn't proceed further from a simulation or a calculation. I would finally thank my parents for facilitating this entire experience and my friends who gave unquestionable amounts of support to help me complete the project.

This report contains 41 pages and 27 figures. Sections 2 thru 6.2 are a part of the first four-week report. Sections 6.3 thru 8 are a part of the second four-week report.

Anurag Pallaprolu
PHYS1211-I
SRF, IISc, Bengaluru.

2 Introduction

Hello. This is an informal report on the work that I have done in the eight weeks as a Summer Research Fellow under Professor Vijay B. Shenoy at the Center for Condensed Matter Physics, Indian Institute of Science, Bengaluru - 12. Apart from the scientific/academic work done, I have learnt a lot about the process of doing science the right way thanks to Professor Shenoy's scintillating ability to explain any concept using undergraduate or advanced undergraduate physics knowledge. The entire idea of deriving something from first principles and the excitement of self-discovery (which was pretty much unknown to me, thanks to the style of education I was exposed to so far) was something I learnt here specifically and is something that I shall carry forward for the rest of my life. I am not entirely sure whether I have made progress in the direction of doing science correctly, but the eight weeks have given me a direction to think in and to work towards. With the introduction out of the way, let us see what I've done in these eight weeks.

3 The Problem

The title of the report relates to the scenario of a sheet of electrons in a magnetic field and the subsequent quantum mechanical analysis of the distribution of energies in the electrons when the field is on and off. To be more pedantic, we are dealing with a two dimensional electron gas in a magnetic field. The problem was solved in its full glory first by Lev Landau. The following elementary derivation is due to Professor Shenoy(see images below).

If recollected correctly, the zero magnetic field case Hamiltonian for a two dimensional electron gas is given by the standard form

$$\epsilon(\bar{p}) = \frac{p_x^2}{2m} + \frac{p_y^2}{2m}$$

However, when there is a magnetic field applied, say in the direction perpendicular to the two dimensional sheet of electrons, we must modify the canonical quantities p_x and p_y into the corresponding electrodynamic counterparts, $p_x - \frac{eA_x}{c}$ and $p_y - \frac{eA_y}{c}$. It is clear that we must know the vector potential inorder to construct these terms. We can choose one of the two commonly used vector potential forms for a given magnetic field. This process of fixing a form for $\bar{A}$ is called choosing a 'gauge'. The two standard gauges available to us are the **symmetric gauge**

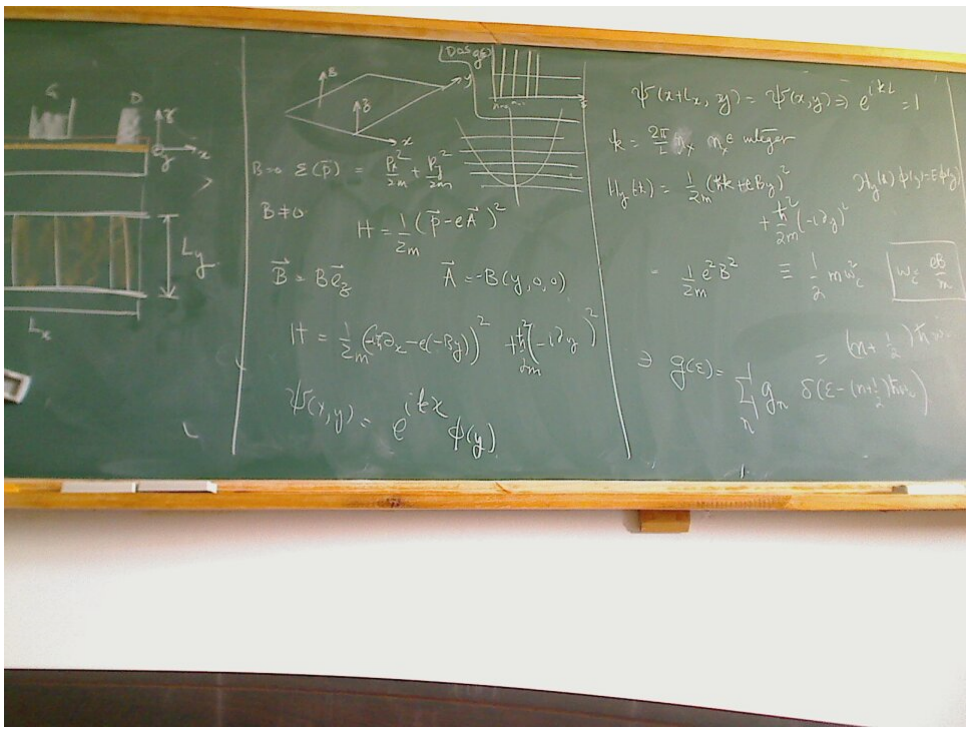
$$\bar{A}_s = \frac{B}{2}(-y, x, 0)$$

and the commonly used gauge for the current situation, the **Landau gauge**

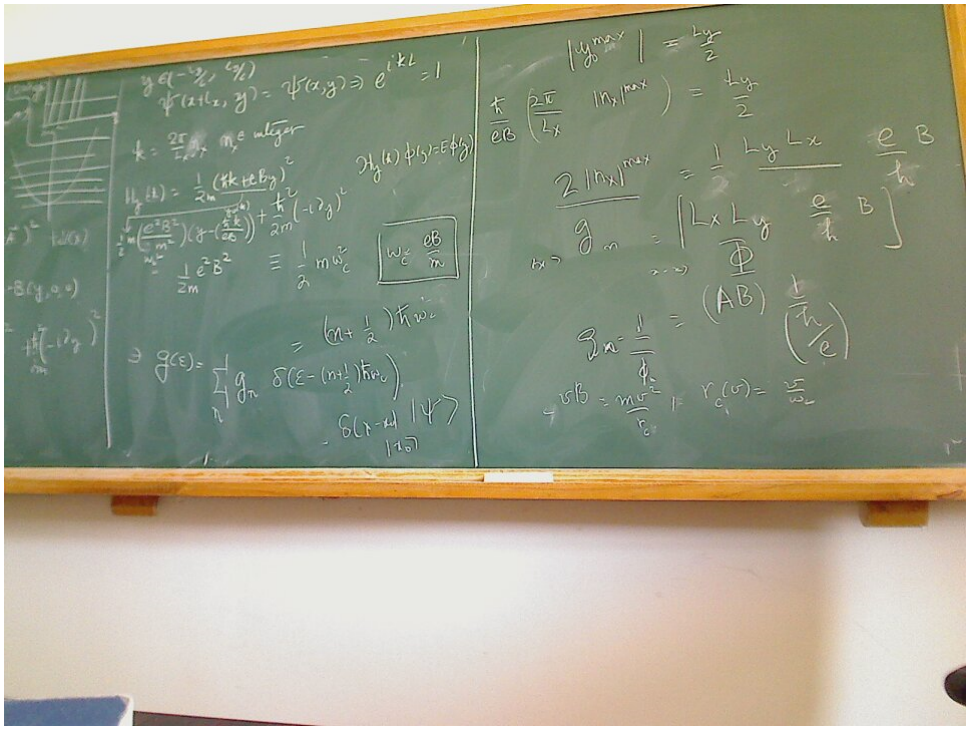
$$\bar{A}_l = -B(y, 0, 0)$$

Choosing the latter case, and setting c = 1, we get the following new Hamiltonian for the Landau problem

$$\mathcal{H} = \frac{1}{2m}(-i\hbar\partial_x + eBy)^2 + \frac{\hbar^2}{2m}(-i\hbar\partial_y)^2$$



Board 1



Board 2

Solving the Schrodinger equation with this as the Hamiltonian gives us the eigenvalues or the energy levels which are none other than the famously known **Landau levels**. A simple way out to solve the Schrodinger equation is to observe that it is nothing but a superposition of the free particle motion in the x direction and of a shifted harmonic oscillator like motion in the y direction. We go with the *ansatz* that the solution looks like

$$\psi(x,y) = e^{ikx}\phi(y)$$

To simplify matters, let us assume a periodic boundary condition in the x direction, say

$$\begin{aligned}\psi(x,y) &= \psi(x+L_x,y)\\ \implies e^{ikL_x} &= 1\\ \implies k &= \frac{2\pi}{L_x}n_x\end{aligned}$$

Now, we have our reduced, purely y dependent hamiltonian

$$\mathcal{H}_y(k) = \frac{1}{2m}(\hbar k + eBy)^2 + \frac{\hbar^2}{2m}(-i\partial_y)^2$$

The y motion, as described before is a shifted harmonic oscillator motion which is the reason for the occurrence for the quantized Landau levels. To find the frequency of oscillation, we just compare the quadratic y coefficient to the standard harmonic oscillator quadratic coefficient $\frac{1}{2}m\omega_c^2$. Doing so, we see that

$$\omega_c = \frac{eB}{m}$$

and the Landau levels are given by

$$\epsilon_{n,L} = (n+\frac{1}{2})\hbar\omega_c$$

The c in the subscript of the angular frequency is due to the fact that this is the same frequency of electrons in a *cyclotron*. Now, enumerating the combined density of states function is easy, as there are quantized levels and no other energies accessible in between,

$$g(\epsilon) = \sum_n g_n\delta(\epsilon - (n+\frac{1}{2})\hbar\omega_c)$$

From the diagram and also from the idea of a harmonic oscillator, we can see that the maximum amplitude of oscillation when started from origin is given by

$$\begin{aligned}|y_{max}| &= \frac{L_y}{2}\\ \frac{\hbar k}{eB} &= \frac{L_y}{2}\\ \frac{\hbar}{eB}(\frac{2\pi}{L_x}n_x) &= \frac{L_y}{2}\\ g_n = 2n_x = L_xL_y\frac{eB}{h} &= (L_xL_yB)\frac{1}{\frac{h}{e}}\end{aligned}$$

Notice that $\frac{h}{e}$ is the quantum elementary flux value a field can have and the first bracket represents the total flux through the sheet. Hence, we come to the startling conclusion that the flux through the sheet is quantized for every Landau level.

$$g_n = \frac{\Phi}{\Phi_e}$$

$$g(\epsilon) = \sum_n g_n \delta(\epsilon - (n + \frac{1}{2})\hbar\omega_c) = \frac{\Phi}{\Phi_e} \sum_n \delta(\epsilon - (n + \frac{1}{2})\hbar\omega_c)$$

For the sake of completeness, I would like to include a few observations/questions that I have made/questioned during the discussion, sort of a commentary,

1 The obvious extension to the Landau problem is the Landau problem with an external electric field. One approach that I wondered is by treating the potential as 'weak' and applying pertubation theory to get the new energy levels and wavefunctions. If one remembers, I had started with the *ansatz* but never really derived the correct structure of the wavefunctions for the situation.

2 A program which was suggested by Professor Shenoy, only to be solved by him later on, was to construct a Gaussian wavepacket and observe its evolution under the Landau situation. The aim of this program was to show that under the classical-quantum correspondence the wavepacket evolution would reproduce the cyclotron orbits one of observe if the Landau situation was treated completely classically. By classical treatment of the Landau problem, I mean, the problem of a 'gas' of electrons in a magnetic field perpendicular to them. Solving the Lorentz equations of motion, one can see that the motion classically is a cyclotron motion, with the radius

$$\frac{mv^2}{r_c} = evB$$

$$\implies r_c = \frac{mv}{eB}$$

and with the solutions looking like

$$x = X_0 + r_c \cos\omega_c t$$

$$y = Y_0 - r_c \sin\omega_c t$$

3 One should be wary of the difference between the canonical momentum p_j and the so called kinetic/electrodynamic momentum $\pi_j = p_j - \frac{eA_j}{c}$. The former quantities always commute whereas the latter do not commute, or commute when there is no magnetic field applied, in which case both the quantities coincide. The commutator for the latter quantities can be worked out and it is easy to see that

$$[\pi_x, \pi_y] = \alpha B_z$$

4 The choice of the gauge is not arbitrary. I chose the Landau gauge for it conserved translational symmetry in one of the dimensions while destroying

rotational symmetry. If the same procedure is followed with the symmetric gauge, one would notice that the angular momentum in the z direction is conserved, implying that the problem is now rotationally symmetric. Again, pardon me for being pedantic, but the symmetric gauge is sometimes referred to as the rotationally invariant gauge in literature.

4 The Wavefunction

In this section, I would like to discuss what the wavefunction $\psi(x, y)$ looks like. Apart from numerous ways in which a multitude of authors had discussed elsewhere in the literature, I had set myself a task to determine the structure of the wavefunctions and I shall discuss my attempt at the problem. All in all, we are dealing with a partial differential equation involving three independent variables, the x and the y coordinates and the time axis.

$$\frac{-\hbar^2}{2m}(\partial_x^2 + \partial_y^2)\psi(x, y, t) + (\frac{e^2B^2y^2}{2m} - 2i\hbar eBy\partial_x)\psi(x, y, t) = i\hbar\frac{\partial\psi}{\partial t}$$

If it isn't clear by now, I'm still using the Landau gauge. Now, one very well known technique for solving partial differential equations is that of the Fast Fourier Transform. The advantage of this technique is plain and simple, it is made for computational execution. To demonstrate, let us solve the Poisson equation using FFT. As is known,

$$(\partial_x^2 + \partial_y^2)\phi(x, y) = -\rho(x, y)$$

Now, the first step is to convert this equation to its equivalent form in a discretized domain. That is, we use the $1 - 1 - 1 - 1 - (-4)$ stencil(see image below) for the second derivatives and the standard forward difference form for the first derivatives(which are absent in this case) and we see that

$$(\partial_x^2 + \partial_y^2)\phi(x, y) = -\rho(x, y)$$

$$\implies \frac{1}{L^2}(\phi_{j+1,k} - 2\phi_{j,k} + \phi_{j-1,k} + \phi_{j,k+1} - 2\phi_{j,k} + \phi_{j,k-1}) = -\rho_{j,k}$$

$$\implies \frac{1}{L^2}(\phi_{j+1,k} + \phi_{j-1,k} + \phi_{j,k+1} + \phi_{j,k-1} - 4\phi_{j,k}) = -\rho_{j,k}$$

Now, we abstractly define the discrete Fourier transform of ϕ as

$$\tilde{\phi}_{m,n} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} W^{mj+nk}\phi_{j,k}$$

$$\implies \phi_{j,k} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W^{-jm-kn}\tilde{\phi}_{m,n}$$

This is nothing special. We are just expanding ϕ in a functional basis and writing it down as a nice double summation for there are two indices. In our problem, we would have three indices, and hence a triple summation. Similarly, for the source term

$$\rho_{j,k} = \frac{1}{N}\sum_{m}^{N-1}\sum_{n}^{N-1} W^{-jm-kn}\tilde{\rho}_{m,n}$$

Plugging this into our difference equation, we get

$$\frac{1}{L^2}[\frac{1}{N}\sum_m\sum_n(W^{-(j+1)m-kn}+W^{-(j-1)m-kn}+W^{-jm-(k+1)n}+W^{-jm-(k-1)n}-4W^{-jm-kn})\tilde{\phi}_{m,n}]$$

$$=$$

$$\frac{1}{N}\sum_m\sum_n W^{-jm-kn}\tilde{\rho}_{m,n}$$

On clearing terms and making $\tilde{\phi}_{m,n}$ the subject, we see that

$$\tilde{\phi}_{m,n}=\frac{L^2\tilde{\rho}_{m,n}}{4-W^m-W^{-m}-W^n-W^{-n}}$$

Now, this can be inverted to retrieve the original function ϕ, and this can be done by a computer with the suitable software package. But the advantage here is, we can see the form of the actual solution by observing the DFT. A similar motivation led me to try the same procedure with the Schrodinger equation of the Landau problem. To restate the equation,

$$\frac{-\hbar^2}{2m}(\partial_x^2+\partial_y^2)\psi(x,y,t)+(\frac{e^2B^2y^2}{2m}-2i\hbar eBy\partial_x)\psi(x,y,t)=i\hbar\frac{\partial\psi}{\partial t}$$

Discretizing the equation using the same stencil

$$\frac{-\hbar^2}{2mL^2}(\psi_{a+1,b,c}+\psi_{a-1,b,c}+\psi_{a,b+1,c}+\psi_{a,b-1,c}-4\psi_{a,b,c})+\frac{e^2B^2b^2L^2}{2m}\psi_{a,b,c}-2i\hbar eBbL\frac{\psi_{a+1,b,c}-\psi_{a,b,c}}{L}$$

$$=$$

$$i\hbar\frac{\psi_{a,b,c+1}-\psi_{a,b,c}}{\tau}$$

Rearranging the terms according to the index,

$$\psi_{a+1,b,c}[\frac{-\hbar^2}{2mL^2}-2i\hbar eBb]+(\psi_{a-1,b,c}+\psi_{a,b+1,c}+\psi_{a,b-1,c})[\frac{-\hbar^2}{2mL^2}]$$

$$+\psi_{a,b,c}[\frac{2\hbar^2}{mL^2}+\frac{e^2B^2b^2L^2}{2m}+2i\hbar eBb+\frac{i\hbar}{L}]-\frac{i\hbar}{L}\psi_{a,b,c+1}=0$$

Let us take the discrete Fourier transform of $\psi_{a,b,c}$.

$$\psi_{a,b,c}=\frac{1}{N}\sum_\alpha\sum_\beta\sum_\gamma W^{-a\alpha-b\beta-c\gamma}\tilde{\psi}_{\alpha,\beta,\gamma}$$

Before directly substituting into the difference equation, note that terms like $b\psi_{a,b,c}$ and $b^2\psi_{a,b,c}$ can be related to $\psi_{a,b,c}$ as derivatives with respect to α,β,γ.

$$b\psi_{a,b,c}=-\frac{\partial\psi_{a,b,c}}{\partial\beta}$$

$$b^2\psi_{a,b,c}=\frac{\partial^2\psi_{a,b,c}}{\partial\beta^2}$$

Conversely, we can see that

$$b\tilde{\psi}_{\alpha,\beta,\gamma}=\frac{\partial\tilde{\psi}_{\alpha,\beta,\gamma}}{\partial\beta}$$

$$b^2\tilde{\psi}_{\alpha,\beta,\gamma} = \frac{\partial^2\tilde{\psi}_{\alpha,\beta,\gamma}}{\partial\beta^2}$$

Plugging the DFT into the difference equation, we end up with a second order *ordinary* differential equation:

$$\frac{d^2\tilde{\psi}_{\alpha,\beta,\gamma}}{d\beta^2} + \frac{4i\hbar m W^\alpha}{eBL^2}\frac{d\tilde{\psi}_{\alpha,\beta,\gamma}}{d\beta} + \frac{2m}{e^2B^2L}\tilde{\psi}_{\alpha,\beta,\gamma} = 0$$

Now, it is clear that the wavefunction will have an exponential structure along with a constant factor which would depend purely on α and γ. The best way to check the correctness of a procedure is to work out the same problem in a different(I would call *this* the standard procedure, due to Hitoshi Murayama of UC, Berkeley)procedure. We construct the raising and lowering ladder operators just as how we would for a standard harmonic oscillator except we replace the canonical quantities with the kinetic/electrodynamic counterparts. For the sake of variety, I also use the symmetric gauge instead of the Landau gauge. The answer, must, anyhow, be independent of this choice. Lowering below the ground state is not possible, for it is the ground state. This implies

$$a|0>=0$$

$$\pi_x + i\pi_y|>=0$$

$$(\partial_x + i\partial_y) - (-y + ix)|0>=0$$

Separating the real and imaginary parts, we see that

$$\frac{\partial|0>}{\partial x} + y|0>=0$$

$$\frac{\partial|0>}{\partial y} - x|0>=0$$

Notice that this relationship holds for whatever x and y coordinates chosen. Now, expand the ground state ket as a Taylor series around infintesimals at the origin.

$$|0>(\epsilon,\delta) = |0>(0,0) + \frac{\partial|0>}{\partial x}_{(\epsilon,\delta)}\epsilon + \frac{\partial|0>}{\partial y}_{(\epsilon,\delta)}\delta + \frac{\partial^2|0>}{\partial x^2}_{\epsilon,\delta}\frac{\epsilon^2}{2!} + \frac{\partial^2|0>}{\partial y^2}_{\epsilon,\delta}\frac{\delta^2}{2!} + ...$$

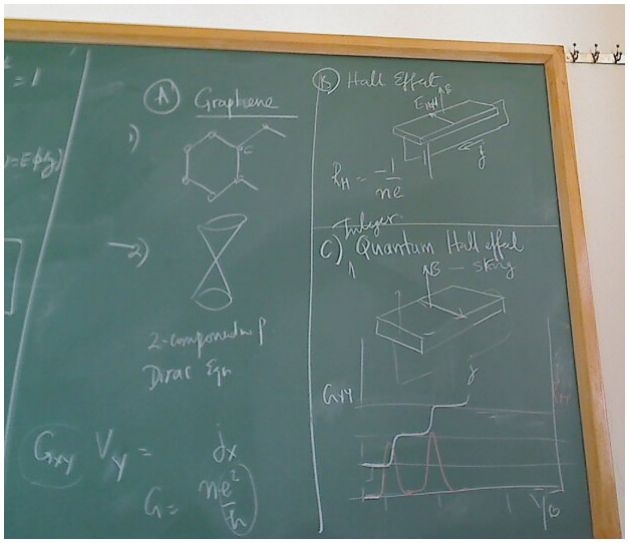
Using the above two relations, we see that the first order derivative terms cancel out and the second order terms can be grouped together as,

$$|0>(\epsilon,\delta) = |0>(0,0)(1 + \epsilon^2\delta^2 + ..)$$

According to Professor Murayama, who solves the same ground state equation by switching to complex coordinates, the solution appears to be of the form

$$\psi(z,\bar{z}) = f(z)e^{\frac{-eBz\bar{z}}{4\hbar c}}$$

and when expanded into a power series, with the corresponding coordinate switches($z \to x + iy$)we see that we have reached at the same conclusion.



Board 3

5 Graphene From Scratch

The next issue I had to tackle was to apply the physics discussed before to this new material called Graphene. In simplest of terms, Graphene is a two dimensional hexagonal arrangement of carbon atoms. I could be less pedantic, in which case, Graphene would be a layer of Graphite who's thickness would be of a single carbon atom. This material was supposedly unstable and was predicted to be non existent at non zero temperatures by the Mermin-Wagner-Hohenberg theorem, or much more precisely, the Landau-Peierls-Mermin-Wagner-Hohenberg-Coleman theorem. But, as it turned out, Andre Geim and Konstantin Novoselov were able to contain Graphene stably and win the Nobel Prize, the details of which are available on the internet. The reason why the substance exists is because, it was found later on that Grpahene is perfecly flat and stable only at 0K and always shows ripples on the surface due to thermal fluctuations thereby intrinsically making it a three dimensional object, than a purely two dimensional one.

The surprising distinction between Graphene and a standard kit of two dimensional electron gas is that the 2DEG follows a quadratic *dispersion* relation, that is, the energy-momentum relationship is quadratic, while in the case of Graphene, it is noted that the dispersion relation is linear when we are dealing with distances close to the surface of the two dimensional lattice. What quantities have a linear dispersion relationship in everyday life? Massless particles do. Familiar examples include Photons and Neutrinos. Now, we are dealing with electrons, which are massive *fermions* and they seem to obey a linear dispersion relation. Fermions and linear dispersion relations have only one thing to suggest, the Dirac equation. Physically, the electrons are no longer governed by a scalar wavefunction, but instead, a four component one.

To proceed any further, I first need a complete understanding of how Graphene looks and behaves. In solid state terms, Graphene is actually made up of two triangular sub-lattices. To show that the energy dispersion is actually linear, we employ a certain approximation in calculating it. It is called the tight binding approximation and is rather self explanatory. I like the explanation given by Feynman in his third volume of the eponymous series and have borrowed the image below from the same source available online for free(*www.feynmanlectures.caltech.edu/*).

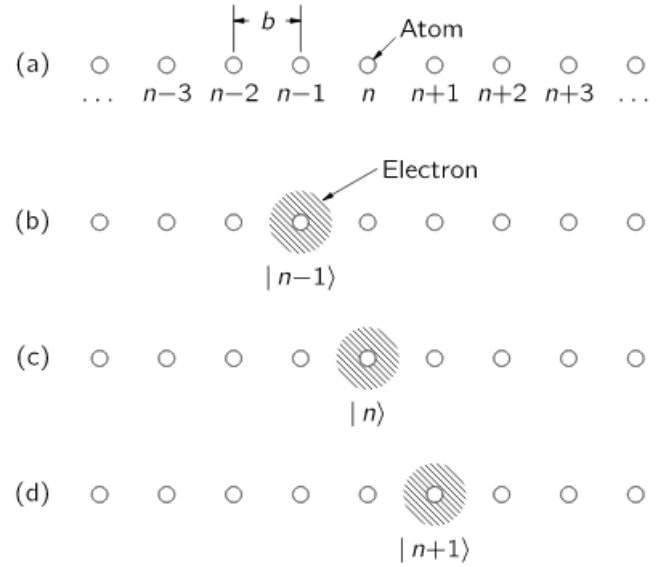


Fig. 13–1. The base states of an electron in a one-dimensional crystal.

The base assumption of the approximation is the idea that the electron has interaction limited to the nearest neighbor atoms. For the linear lattice, one can write the following series of Schrodinger differential equations, assuming the electron has equal amplitude to move to $|n-1>$ and $|n+1>$.

$$i\hbar\frac{dC_{n-1}}{dt} = E_0C_{n-1} - AC_{n-2} - AC_n$$

$$i\hbar\frac{dC_n}{dt} = E_0C_n - AC_{n-1} - AC_{n+1}$$

$$i\hbar\frac{dC_{n+1}}{dt} = E_0C_{n+1} - AC_n - AC_{n+2}$$

These can be solved, either iteratively (requires a computer), or using an *ansatz* solution for one of the three differential equations. The latter approach leads us to writing

$$C_n = e^{i(kx_n - \frac{Et}{\hbar})}$$

Although these approximations look naive, for a first guess, this is the best a human could do. Now, plugging this into the second of the three I've stated before, it is easy to see that the dispersion relation turns out to be sinusoidal and, to be a bit more precise, of the form

$$E(k) = E_0 - 2A\cos(kb)$$

This can be easily verified with the fact that $x_{n+1} = x_n + b$ and $x_{n-1} = x_n - b$. I now want to apply this procedure for the Graphene lattice. Again, technically we are trying to locate the tight binding hamiltonian for a two dimensional "honeycomb" lattice. Just to clarify, in the linear lattice case, the tight binding hamiltonian was the right hand side, that is,

$$\hat{\mathcal{H}}|n>= E_0|n> -A|n-1> -A|n+1>$$

where a more technical name for A would be the hopping parameter or the hopping integral, which would play a decently significant role in Graphene's tight binding hamiltonian. A look at the image below, from the Physics Wiki of The University of Delaware:

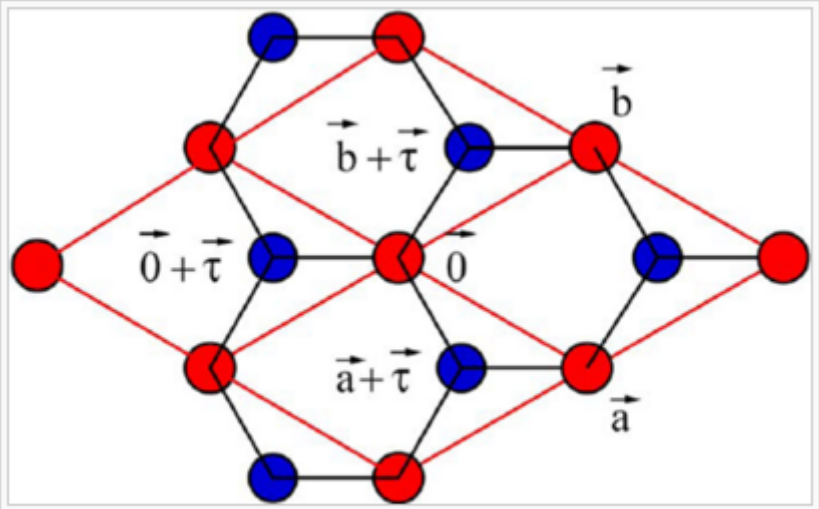
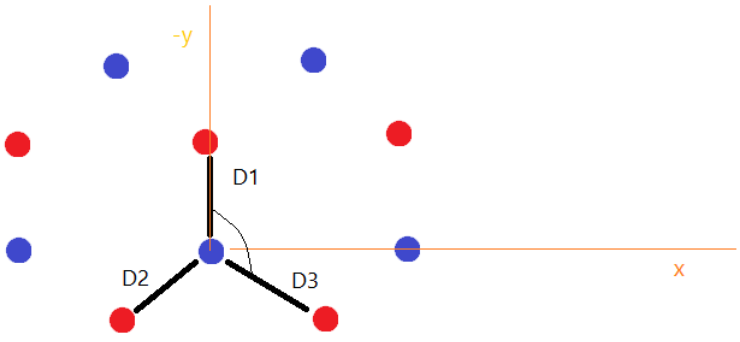


FIG 3: Vectors specifying the position of carbon atoms on two triangular Bravais sublattices comprising the honeycomb lattice of graphene.

should motivate a similar definition for the tight binding hamiltonian. There might be some confusion as to why the three carbons in blue which look symmetric in the image have different vector labels. When I redrew the image, it became clearer



If the interatomic spacing is taken to be a, then from vector algebra and trigonometry, it is easy to see that

$$\bar{D}_1 = -a\hat{y}$$

$$\bar{D}_2 = \frac{a}{2}(-\sqrt{3}\hat{x} + \hat{y})$$

$$\bar{D}_3 = \frac{a}{2}(\sqrt{3}\hat{x} + \hat{y})$$

This should essentially make the point clear that, although, lattice-wise they look similar, they are actually three distinct atoms. But, they can be labeled by two primitive vectors $\bar{a}$ and $\bar{b}$ as there are two trigonal primitive sublattices and by a third vector called $\bar{\tau}$ which signifies the "hopping" distance between two atoms of different sublattices. With this labeling out of the way, constructing the actual tight binding hamiltonian isn't really that hard. All we need to do is consider all possible nearest neighbor hoppings and assign the hopping integral, say Λ as the amplitude for the process. In the image from the University of Delaware, one can see that the central carbon atom is labeled as $\bar{0}$. For the most general hamiltonian, we take this atom to be at a lattice vector $\bar{R}$ and sum over all possible vectors. For each hop onto a neighboring atom, as we discussed before, we assign an amplitude of Λ, and we construct the following:

$$\hat{H} = -\Lambda \sum_{\bar{R}} (|\bar{R}><\bar{R}+\tau| + |\bar{R}><\bar{R}+\bar{a}+\tau| + |\bar{R}><\bar{R}+\bar{b}+\tau| + \epsilon)$$

where ϵ is the harmonic conjugate(write the outer brackets in the reverse order and add to the same term)of the previous terms inside the summation. Getting the dispersion relation is not much of an effort as we have to compute the matrix element of an electron hopping from one sublattice to the other. If I label the state ket of the electron on the A sublattice as $|A(k)>$ and the state ket of the electron on the B sublattice as $|B(k)>$, then, I have to calculate $< A(k)|\hat{H}|B(k) >$. The state kets are well known Bloch states expanded in their own wave vector Fourier representation

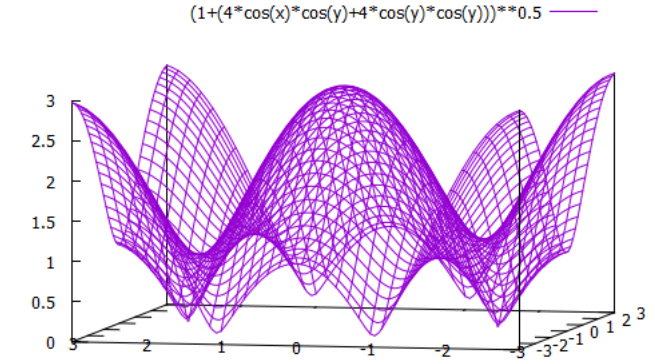
$$|A(k)>= \frac{1}{\sqrt{N}} \sum_{\bar{R}} e^{i\bar{k}.\bar{R}} |\bar{R}>$$

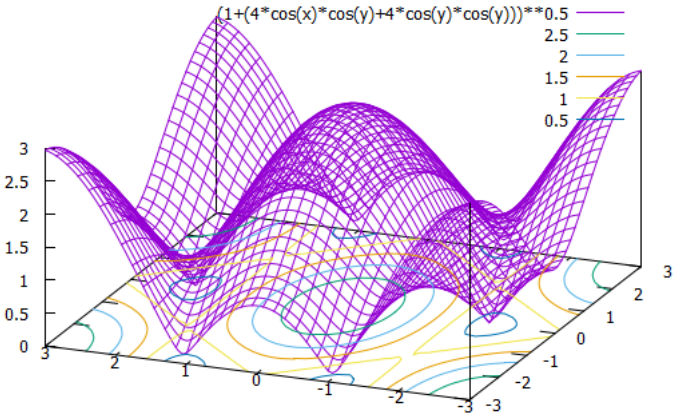
$$|B(k)>= \frac{1}{\sqrt{N}} \sum_{\bar{R}} e^{i\bar{k}.\bar{R}} |\bar{R}+\tau>$$

Plugging these into the matrix element, one can see that

$$| < A(k)|\hat{H}|B(k) > | = E(k_x, k_y, k_z) = \Lambda\sqrt{1 + 4\cos k_x \cos k_y + 4\cos^2 k_y}$$

The graphs below should show how the dispersion relation goes from quadratic to linear as I move closer to the surface of Graphene.


(1+(4*cos(x)*cos(y)+4*cos(y)*cos(y)))**0.5
3
2.5
2
1.5
1
0.5
0
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


(1+(4*cos(x)*cos(y)+4*cos(y)*cos(y)))**0.5
2.5
2
1.5
1
0.5
3
2.5
2
1.5
1
0.5
0
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

6 The Landau Problem in Graphene with an Electric Field

As done with a normal 2DEG, I solve the Landau problem with Graphene. To generalize the procedure, I also include an external electric potential. Again, I solved the problem as a mathematical one than as a problem in physics i.e., work out the matrix equations, end up with a differential equation and then solve it. Professor Shenoy, on the other hand, worked out the same problem completely using a standard technique in mechanical and civil engineering known as the *Finite Element Method*. My mathematical solution ultimately ended up in using a FEM package to solve my PDE(*FreeFEM++*, a C++ extension), although I used with absolutely no idea as to what FEM was. I thank Professor Shenoy for enlightening me about the same, and have included the discussion in this section.

6.1 Mathematical Solution

The standard Dirac hamiltonian is given by the following 4x4 matrix:

$$\begin{pmatrix} -\bar{\sigma}^*.\bar{p} & 0 \\ 0 & \bar{\sigma}.\bar{p} \end{pmatrix}$$

where $\bar{\sigma}$ is the Pauli spin matrix vector and $\bar{p}$ is the canonical momentum. As the usual procedure goes, replace $\bar{p}$ with $\bar{\pi}$ and add a potential term, which is the scalar potential multiplied by the 4x4 identity element.

$$\begin{pmatrix} -\bar{\sigma}^*.\bar{\pi} + V(x)\mathbb{I}_2 & 0 \\ 0 & \bar{\sigma}.\bar{\pi} + V(x)\mathbb{I}_2 \end{pmatrix} \Psi = \epsilon\Psi$$

As stated before, Ψ is a four component wavefunction. Now as to what each component means, I shall explain later, but for now, since I'm focusing on solving the problem mathematically, I state that we can group the four components into two groups of two components each(each component belonging to a sublattice).

$$\Psi = \begin{pmatrix} \tilde{\phi}^1 \\ \tilde{\phi}^2 \end{pmatrix}$$

$$\tilde{\phi}^j = \begin{pmatrix} \phi^j_A \\ \phi^j_B \end{pmatrix}$$

$$\implies \begin{pmatrix} -\bar{\sigma}^*.\bar{\pi} + V(x)\mathbb{I}_2 & 0 \\ 0 & \bar{\sigma}.\bar{\pi} + V(x)\mathbb{I}_2 \end{pmatrix} \begin{pmatrix} \tilde{\phi}^1 \\ \tilde{\phi}^2 \end{pmatrix} = \epsilon \begin{pmatrix} \tilde{\phi}^1 \\ \tilde{\phi}^2 \end{pmatrix}$$

$$[-\bar{\sigma}^*.\bar{\pi} + V(x)\mathbb{I}_2]\tilde{\phi}^1 = \epsilon\tilde{\phi}^1$$

$$[\bar{\sigma}.\bar{\pi} + V(x)\mathbb{I}_2]\tilde{\phi}^2 = \epsilon\tilde{\phi}^2$$

We need only the first two components of the Pauli spin matrix vector

$$\bar{\sigma}^* = [\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -i \\ +i & 0 \end{pmatrix}]$$

I use the symmetric gauge, and writing the first matrix equation in its full expanded form, we see that

$$-[\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}(\hat{p}_x+\frac{eB\hat{y}}{2c}))+\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}(\hat{p}_y-\frac{eB\hat{x}}{2c})+\begin{pmatrix}V(x) & 0\\ 0 & V(x)\end{pmatrix}]\begin{pmatrix}\phi_A^1\\ \phi_B^1\end{pmatrix}=\epsilon\begin{pmatrix}\phi_A^1\\ \phi_B^2\end{pmatrix}$$

Or, adding up the 2x2 matrices, we end up with

$$\begin{pmatrix}V(x) & -[\hat{p}_x-i\hat{p}_y+\frac{eB}{2c}(\hat{y}+i\hat{x})]\\ -[\hat{p}_x+i\hat{p}_y+\frac{eB}{2c}(\hat{y}-i\hat{x})] & V(x)\end{pmatrix}\begin{pmatrix}\phi_A^1\\ \phi_B^2\end{pmatrix}=\epsilon\begin{pmatrix}\phi_A^1\\ \phi_B^2\end{pmatrix}$$

Again, writing the equations separately, we see that

$$V(x)\phi_A^1-(\hat{p}_x-i\hat{p}_y+\frac{eB}{2c}(\hat{y}+i\hat{x}))\phi_B^1=\epsilon\phi_A^1$$

$$V(x)\phi_B^1-(\hat{p}_x+i\hat{p}_y+\frac{eB}{2c}(\hat{y}-i\hat{x}))\phi_A^1=\epsilon\phi_B^1$$

The equations look symmetric and as is suggested, I proceed by elimination, substituting for ϕ_A^1,

$$[\hat{p}_x+i\hat{p}_y+\frac{eB}{2c}(\hat{y}-i\hat{x})][\hat{p}_x-i\hat{p}_y+\frac{eB}{2c}(\hat{y}+i\hat{x})]\phi_B^1=(V(x)-\epsilon)^2\phi_B^1$$

$$\implies(\hat{p}_x+\frac{eB\hat{y}}{2c})^2+(\hat{p}_y-\frac{eB\hat{x}}{2c})^2+i[\hat{p}_y-\frac{eB\hat{x}}{2c},\hat{p}_x+\frac{eB\hat{y}}{2c}]\phi_B^1=(V(x)-\epsilon)^2\phi_B^1$$

where $[,]$ are the standard commutation brackets. Simplifying further

$$(\hat{p}_x^2+\hat{p}_y^2+\frac{e^2B^2}{2c}(\hat{x}^2+\hat{y}^2)+\frac{eB}{c}(\hat{y}\hat{p}_x-\hat{x}\hat{p}_y)+i[\hat{p}_y-\frac{eB\hat{x}}{2c},\hat{p}_x+\frac{eB\hat{y}}{2c}])\phi_B^1=(V(x)-\epsilon)^2\phi_B^1$$

Using the canonical commutation relations,

$$(\hat{p}_x^2+\hat{p}_y^2+\frac{eB}{c}(\hat{y}\hat{p}_x-\hat{x}\hat{p}_y)+(\hat{x}^2+\hat{y}^2)\frac{e^2B^2}{2c}+\frac{\hbar eB}{c})\phi_B^1=(V(x)-\epsilon)^2\phi_B^1$$

This is the (eigenvalue) partial differential equation which we have to solve. I substitute $\hat{p}_x=-i\hbar\partial_x$ and $\hat{p}_y=-i\hbar\partial_y$. Also, using the natural units in which we set $e=c=\hbar=1$, we see that the differential equation is much simplified to

$$(-(\partial_x^2+\partial_y^2)+B(y(-i\partial_x)-x(-i\partial_y))+\frac{x^2+y^2}{2}B^2+B)\phi_B^1=(V(x)-\epsilon)^2\phi_B^1$$

My first attempt at trying to find out the structure of $\phi_B^1(x,y)$ was to use a polynomial series expansion upto quadratic order and figure out the initial coefficients. That is

$$\phi_B^1(x,y)=\lambda+A_1x+B_1y+\mu_1xy+A_2x^2+B_2y^2$$

This is **wrong** as it leads to senseless conclusions from the relationships between various coefficients. According to technical mathematical documentation, I am officially dealing with a **complex elliptic partial differential equation**, and every elliptic PDE can be written in the form of

$$\sum_{i,j}a^{ij}\phi_{ij}+\sum_i b^i\phi_i+c\phi=\lambda\phi$$

In this case, we can configure these coefficients:

$$a^{ij} = -\delta^{ij}$$

$$b^i = (-iBx_2, iBx_1)$$

$$c = c(x_1, x_2) = \frac{x_1^2 + x_2^2}{2}B^2 + B$$

FreeFEM++ is a C++-like programming language used to solve partial differential equations using the finite element method. One does not need explicit knowledge of FEM to use the language, but one must know how to reduce a partial differential equation into its so called *weak form*. How does one do that? Let v be a smooth function in the same domain of definition as ϕ. Note the following transformations

$$\sum_{i,j} a^{ij}\phi_{ij} + \sum_i b^i\phi_i + c\phi = \lambda\phi$$

$$\int_{\mathcal{T}} \sum_{i,j} a^{ij}\phi_{ij}v + \sum_i b^i\phi_i v + c\phi v = \int_{\mathcal{T}} \lambda\phi v$$

Using integration by parts on the first term, we see that this can be reduced to the following, with appropriate boundary conditions

$$\int_{\mathcal{T}} \sum_{i,j} a^{ij}\phi_i v_j + \sum_i b^i\phi_i v + c\phi v = \int_{\mathcal{T}} \lambda\phi v$$

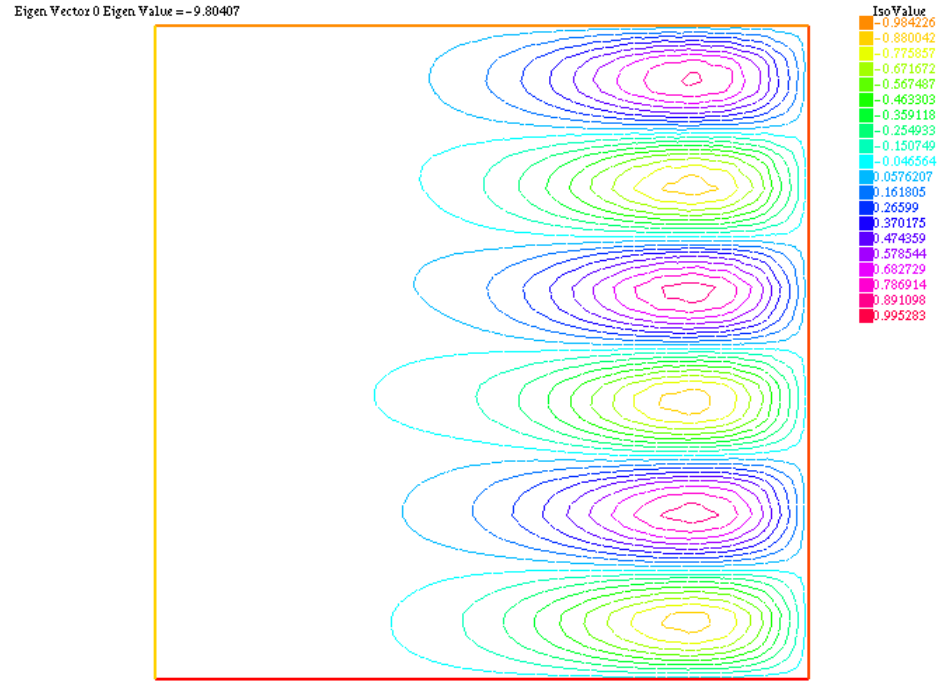
This is supposedly the weak form of the elliptic PDE. The FreeFEM++ PDE solver works on this weak form and hence, we should convert our PDE to this form and then feed it into FreeFEM++. I once again notify that none of this is relevant to how FEM happens(which is the subject for the next section). Before any of this, the above theory applies to real elliptic PDEs and not complex ones. Hence, taking the modulus of the PDE on both sides, we get the following eigenvalue equation for the **modulus** of the wavefunction. Also, the eigenvalues are, from comparison, $(V(x) - \epsilon)^2$.

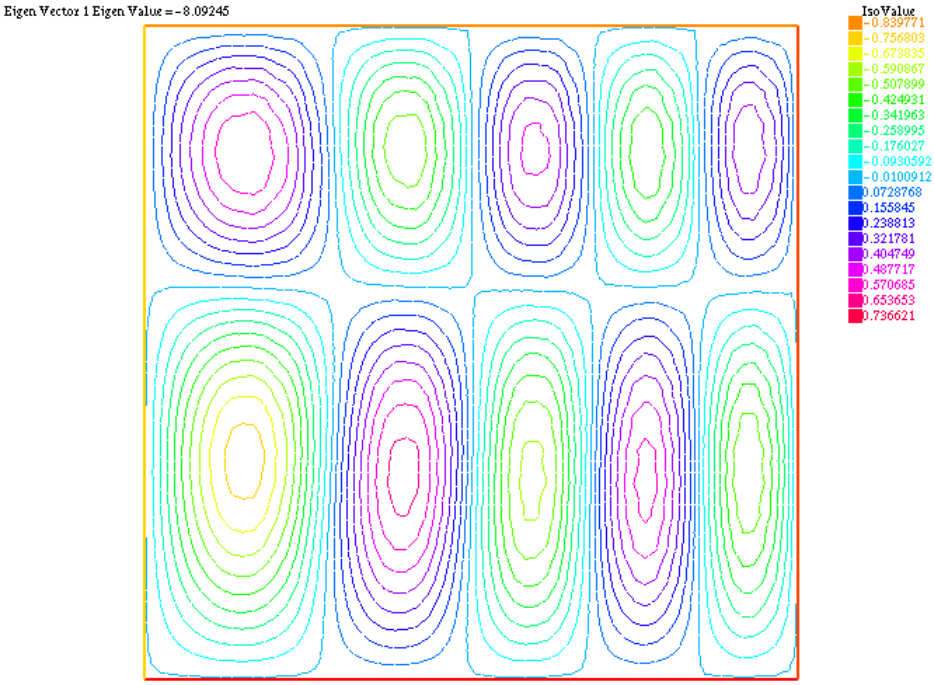
$$(-(\partial_x^2 + \partial_y^2) - B(y(\partial_x) - x(\partial_y)) + \frac{x^2 + y^2}{2}B^2 + B)|\phi_B^1| = (V(x) - \epsilon)^2|\phi_B^1|$$

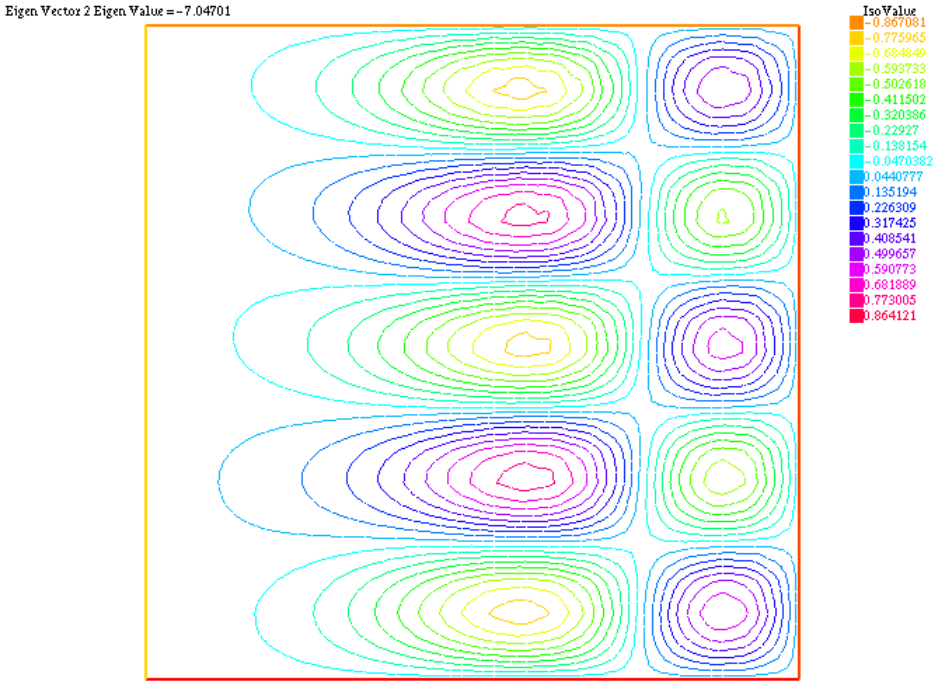
The FreeFEM++ simulation computes the first twenty eigenvalues from the ground state, with candidate value of $B = 1T$ on a square domain. The color plots are those of $|\phi_B^1|$.

```
verbosity=10;
mesh Th=square(20,20,[pi*x,pi*y]);
fespace Vh(Th,P2);
Vh u1,u2;
func f=sin(x*y);
real sigma = 20; // value of the shift
```

```
varf op(u1,u2)= int2d(Th)( - dx(u1)*dx(u2) - dy(u1)*dy(u2) - y*dx(u1)*u2 + x*dy
- sigma*u1*u2 )+ on(1,2,3,4,u1=0) ; // Boundary condition
varf b([u1],[u2]) = int2d(Th)( u1*u2 )
matrix OP= op(Vh,Vh, solver=Crout, factorize=1);
matrix B= b(Vh,Vh, solver=CG, eps=1e-20);
int nev=20; // number of computed eigen value close to sigma
real[int] ev(nev); // to store the nev eigenvalue
Vh[int] eV(nev); // to store the nev eigenvector
int k=EigenValue(OP,B,sym=true, sigma=sigma, value=ev, vector=eV,
tol=1e-10,maxit=0,ncv=0);
```


Eigen Vector 0 Eigen Value = -9.80407
IsoValue
-0.984226
-0.880042
-0.775857
-0.671672
-0.567487
-0.463303
-0.359118
-0.254933
-0.150749
-0.046564
0.0576207
0.161805
0.26599
0.370175
0.474359
0.578544
0.682729
0.786914
0.891098
0.995283


Eigen Vector 1 Eigen Value = -8.09245
IsoValue
-0.839771
-0.756803
-0.673835
-0.590867
-0.507899
-0.424931
-0.341963
-0.258995
-0.176027
-0.0930592
-0.0100912
0.0728768
0.155845
0.238813
0.321781
0.404749
0.487717
0.570685
0.653653
0.736621



The procedure for identifying the exact structure of the eigenfunctions for the complex PDE is done iteratively rather, for we are now hunting the real and imaginary parts of the wavefunction using one connecting equation. Let us say

$$\phi_B^1 = \phi = \phi_R + i\phi_I$$

Substituting this,

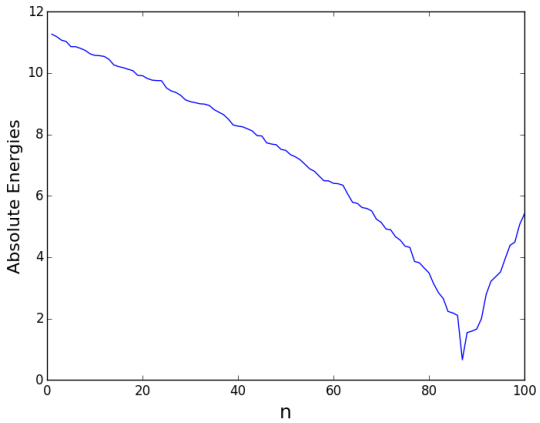
$$(-(\partial_x^2 + \partial_y^2) + (\frac{x^2+y^2}{2}B^2 + B))\phi_R + (B(y(\partial_x) - x(\partial_y))\phi_I = \lambda\phi_R$$

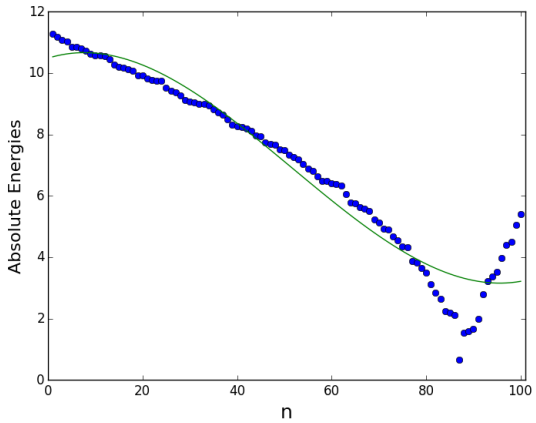
$$(-(\partial_x^2 + \partial_y^2) + (\frac{x^2+y^2}{2}B^2 + B))\phi_I - (B(y(\partial_x) - x(\partial_y))\phi_R = \lambda\phi_I$$

Direct solution of simultaneous eigenvalue elliptic partial differential equations did not look like it was solvable by FreeFEM++, however, the iterative procedure can be seen by setting ϕ_I to be zero initially and running the first equation as a pure eigenvalue equation in ϕ_R and then to solve the second equation for ϕ_I using this first order approximation for ϕ_R.

From numerous technical documents available on Graphene, one can verify that the eigenvalues vary as the square root of the Landau level number n. From our simulation data above, we can see that the curve fitting generates the same solution. We take the absolute values of the negative eigenvalues and plot from quantum number $n = 0$ to $n = 100$. The first image is the smooth fit and the second one shows the actual interpolation. We get negative eigenvalues because FreeFEM++ does not know we're working on a physics problem.

```
[ 11.26578892  11.18762709  11.07348184  11.02769242  10.86043277
  10.85822269  10.80476747  10.73769063  10.62746442  10.57511229
  10.56584119  10.5399241   10.44332323  10.26542741  10.20950537
  10.17388815  10.12151174  10.07754931   9.9259055    9.91359168
   9.82456615   9.76935515   9.75583928   9.74982564   9.51821412
   9.41844467   9.36682444   9.27351605   9.12630265   9.06996692
   9.03639309   8.99840541   8.98783066   8.94223686   8.80872863
   8.72476361   8.64164915   8.49601671   8.30800818   8.27047157
   8.2452168    8.18213297   8.11343331   7.96170836   7.94917606
   7.72657104   7.68709958   7.66630289   7.51521124   7.47990642
   7.34198202   7.2722555    7.17713035   7.03386096   6.8805741
   6.80077201   6.64456169   6.49077807   6.48689448   6.40595816
   6.39264421   6.34394987   6.05446942   5.78968047   5.75438094
   5.61728582   5.58835396   5.50764923   5.23946562   5.13415037
   4.92089423   4.89024539   4.66682976   4.5542288    4.35985091
   4.32050923   3.86011658   3.81865159   3.64953422   3.49200515
   3.13114516   2.84472319   2.6546205    2.23233286   2.18468991
   2.10464249   0.65718034   1.54434776   1.59089283   1.65793546
   1.99626902   2.79268688   3.21540044   3.36616993   3.51992898
   3.96378607   4.38943049   4.48912018   5.06374367   5.40705095]
```





6.2 The Finite Element Method

The above computation which I have done, is based on this method. In this aspect, FreeFEM++ is a very good package for end users as one does not need to know what FEM does, and it has inbuilt routines to compute eigenvalues for comparatively large grid sizes. Before I dive into what the method actually is, a few technicalities. Our Hamiltonian, or atleast an element looks like

$$\hbar\tilde{v}_F\bar{\sigma}.\bar{\pi} + \mathcal{V}(x)\mathbb{I}$$

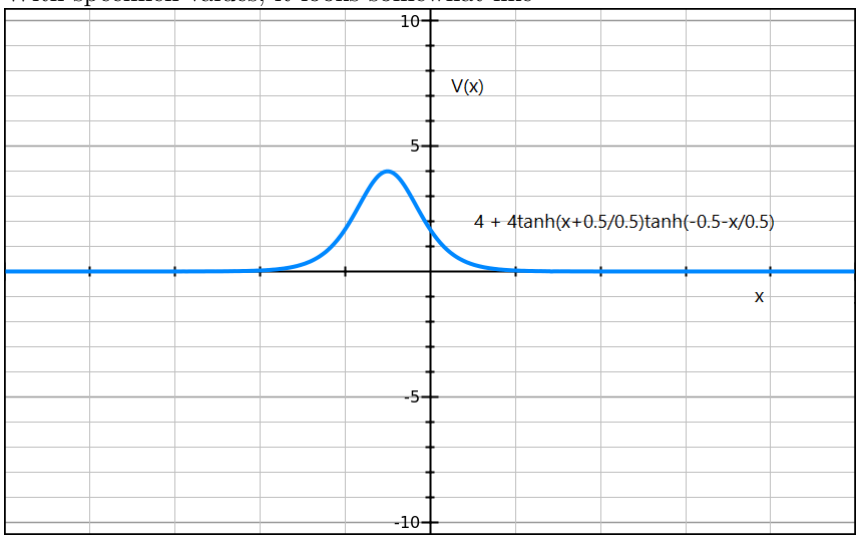
where the constant $\tilde{v}_F$ is the Fermi velocity. Using the Landau gauge, we see that

$$[\hbar\tilde{v}_F\bar{\sigma}.(-i\bar{\partial} - eBy\hat{e}_x) + \mathcal{V}(x)\mathbb{I}]\Psi = E\Psi$$

Here Ψ is the two component wavefunction and **not** the scalar one. Note that it is a 2X2 system on both sides. The next technicality involves the structure of $\mathcal{V}(x)$ as a modulated step function.

$$\mathcal{V}(x) = \frac{V_0}{2} + \frac{V_0}{2}(tanh(\frac{x + L_G}{\zeta_0})tanh(\frac{L_G - x}{\zeta_0}))$$

With specimen values, it looks somewhat like



I define a new equivalent potential term

$$\mathbb{V}_B(x) = -\hbar\tilde{v}_F eBy\sigma_x + \mathcal{V}(x)\mathbb{I}$$

Now, our effective hamiltonian looks like

$$\mathcal{H}\Psi = [\hbar\tilde{v}_F\bar{\sigma}.(-i\bar{\partial}) + \mathbb{V}_B(\bar{r})]\Psi = \lambda\Psi$$

This is a field equation(the Dirac field). If recalled correctly, given a Lagrangian density one can construct the field equations by minimizing the effective action. The resulting equations are known as the Euler-Lagrange equations corresponding to the density. The idea here is to work backwards instead, and take this field equation as your Euler-Lagrange equation and look for a suitable Lagrangian density. This process won't assure a unique density, for, from a standard theorem, if I add the gradient of a smooth scalar function to a given L density, the resulting Euler-Lagrange equations would be invariant. Luckily, we can construct the density, and hence the action for the Dirac field.

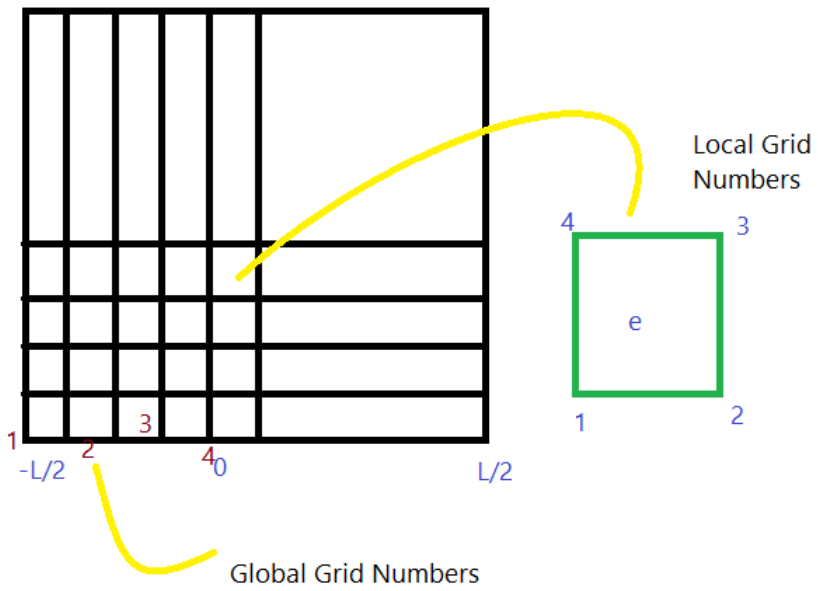
$$\epsilon[\Psi^*, \Psi; \lambda] = \int_\Omega d^2\bar{r}\Psi^*(\bar{r})[\mathcal{H}(\bar{r})]\Psi(\bar{r}) - \lambda\Psi^*(\bar{r})\Psi(\bar{r})$$

If I extremize this, I would get back the original field equation. For simplicity, I relabel

$$\mathcal{H}(\bar{r}) - \lambda\mathbb{I} = \tilde{\mathcal{H}}(\bar{r})$$

$$\epsilon[\Psi^*, \Psi; \lambda] = \int_\Omega d^2\bar{r}\Psi^*(\bar{r})\tilde{\mathcal{H}}(\bar{r})\Psi(\bar{r})$$

With these apparent technicalities out of the way, let us look at the domain which we are working on. It is a square domain with a square mesh on it. Each element in the square mesh is the "finite element" in the finite element method.



The nodes in the square domain are numbered using a **global numbering** scheme. Each square element has a separate indexing for its nodes known as the **local numbering** scheme. It is straightforward to see, that for a given element e, and a given local node number, one can easily compute the global node number of the corresponding node. We symbolize this with the help of

$$j = gn(e, i)$$

where gn is known as the **connectivity matrix**. The advantage of the finite element method is the following: *given an element, and the values of the wavefunction at the four corners, I can calculate the value of the wavefunction at any other point inside the element.* Mathematically

$$\Psi^e(\bar{r}) = \sum_{i \in e} \Psi_i^e(\bar{r}) N_i^e(\bar{r})$$

The new functions $N_i^e(\bar{r})$ are standard weighting functions known as **shape functions** which are completely mesh dependent and are different for different local nodes. I will show the exact structure of these functions in a while. Getting back to our action functional, we can write the action for the entire domain as the sum of these finite element actions:

$$\epsilon[\Psi^*, \Psi; \lambda] = \sum_e \int_{\Omega_e} d^2\bar{r} \Psi^{e*}(\bar{r}) \tilde{H}(\bar{r}) \Psi^e(\bar{r})$$

$$= \int_\Omega \sum_e d^2\bar{r}\Psi^{e*}(\bar{r})\tilde{H}(\bar{r})\Psi^e(\bar{r})$$

$$= \int_\Omega \sum_e d^2\bar{r} \sum_{i\in e} \Psi_i^{e*}(\bar{r})N_i^e(\bar{r})\tilde{H}(\bar{r}) \sum_{i'\in e} \Psi_{i'}^e(\bar{r})N_{i'}^e(\bar{r})$$

$$= \sum_e \sum_{ii'} \int d^2\bar{r}\Psi_i^{e*}(\bar{r})N_i^e(\bar{r})\tilde{H}(\bar{r})N_{i'}^e(\bar{r})\Psi_i^e(\bar{r})$$

We group the central term under the new notation

$$N_i^e(\bar{r})\tilde{H}(\bar{r})N_{i'}^e(\bar{r}) = \tilde{H}_{ii'}^e$$

The right hand side is more commonly known in the engineering sciences as the element of the stress matrix.

$$\epsilon[\Psi^*, \Psi; \lambda] = \sum_e \sum_{ii'} \int d^2\bar{r}\Psi_i^{e*}(\bar{r})\tilde{H}_{ii'}^e\Psi_i^e(\bar{r})$$

To get things back to physics, we are still solving an eigenvalue equation, but now the same eigenvalue equation has been reduced to a smaller, localized domain. Extremizing this form of the action(the finite element form), we see that we have to solve the following

$$(\mathcal{H} - \lambda\mathcal{M}\mathbb{I})\Psi = 0$$

where the matrix $\mathcal{M}$ is seen to be

$$\mathcal{M}_{ii'} = \mathbb{I}\int d^2\bar{r}N_{gn(e,i)}^e(x,y)N_{gn(e,i')}^e(x,y)$$

How do I work out what $\mathcal{H}$ matrix is for the larger square domain? Once again, the connectivity matrix comes to our rescue.

$$\mathcal{H}_{gn(e,i),gn(e,i')} = \mathcal{H}_{i,i'}^e$$

$$H_{i,i'}^e = \int d\bar{r}N_i^e(\bar{r})[\hbar\bar{\sigma}.(-i\bar{\partial}) + \mathbb{V}_B(\bar{r})]N_{i'}^e(\bar{r})$$

$$= \hbar\tilde{v}_F\bar{\sigma}[\int d^2\bar{r}N_i^e(\bar{r})(-i\bar{\partial})N_{i'}^e(\bar{r})] + \sum_k \mathbb{V}_k^e \int d^2\bar{r}N_i^e(\bar{r})N_k^e(\bar{r})N_{i'}^e(\bar{r})$$

where $\mathbb{V}_k^e$ is the value of the modified potential on the elemental nodes. As far as the shape functions are concerned, they are completely mesh shape dependent and are well tabulated. For a square mesh with edge coordinates $(0,0),(0,b),(a,0),(a,b)$, the nodal shape functions are given by

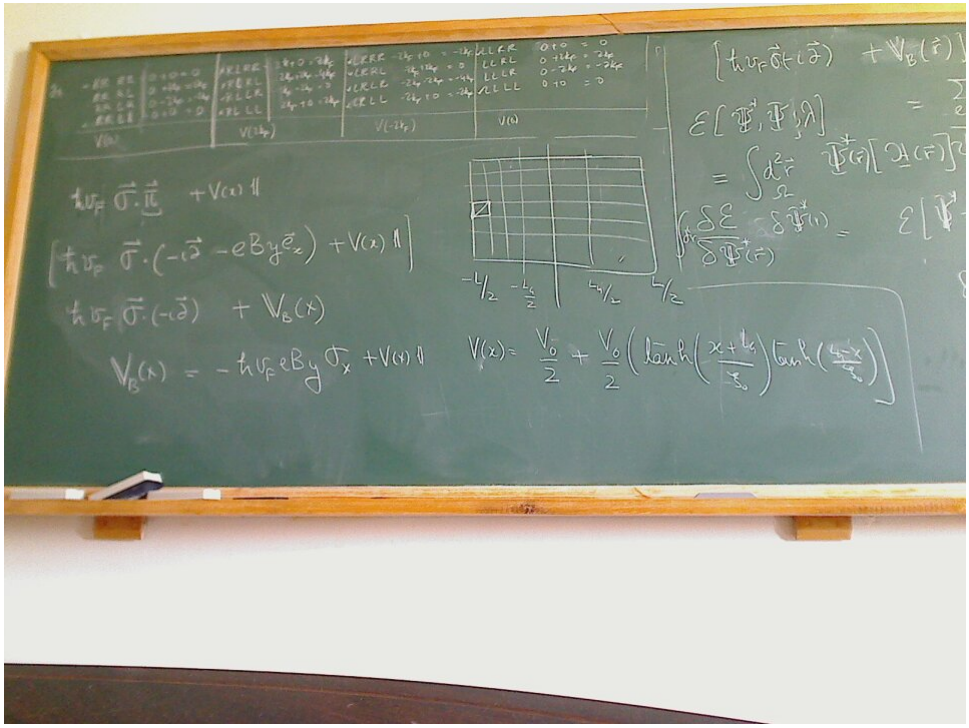
$$N_1(x) = \frac{(x-a)(y-b)}{ab}$$

$$N_2(x) = \frac{-x(y-b)}{ab}$$

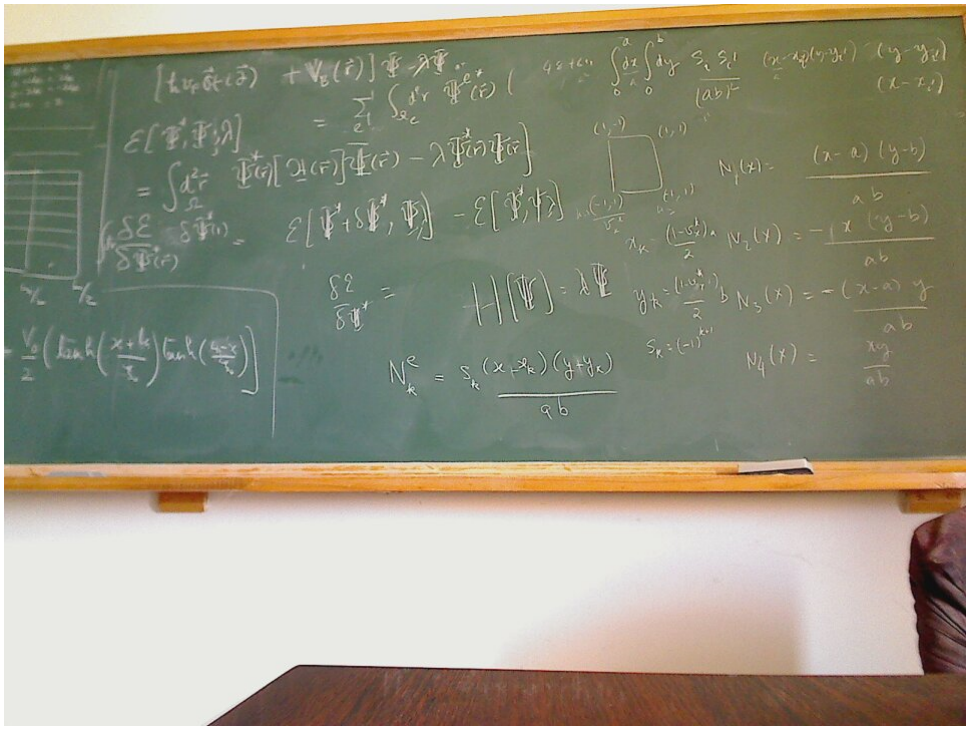
$$N_3(x) = \frac{-(x-a)y}{ab}$$

$$N_4(x) = \frac{xy}{ab}$$

Another advantage of the finite element scheme is apparent. All we need to construct the eigenvalue equation are integrals of these polynomial expressions which can be done by hand easily or by Mathematica routines. I thank Professor Shenoy for taking his time and explaining the procedure to me.



Board 4



Board 5

6.3 The Finite Difference Method

This is an update to the four week report. There is an issue with the finite element scheme, at the low energy or the infrared range. The finite element scheme predicts the existence of four Dirac cones at the four vertices of the square element. This being said, the issue here is again with the fact that the procedure is a mathematical one used by engineers in a wide variety of applications, and it is the physical situation that renders the scheme wrong. We stop working with the linear dispersion relation, and once again get back to analyzing **quadratic** fermions in Graphene. To do this, we need to create a band gap between the electron and hole band structures, and we label this as $\Delta\tau_x$. This is a 2X2 matrix which is numerically equivalent to the Pauli X, but it does **not** represent spin dependence. With this term's inclusion in the hamiltonian, the net band structure looks somewhat like that of a semiconductor. The modified hamiltonian looks like

$$\frac{\hbar^2}{2m}(-i\bar{\nabla}+\frac{e\bar{A}}{\hbar})^2\tau_z+\Delta\tau_x+\mathcal{V}(x)\mathbb{I}$$

On reduction and simplification,

$$\frac{\hbar^2}{2m}[-\partial_x^2-\partial_y^2+\frac{2eByi\partial_x}{\hbar}+\frac{e^2B^2y^2}{\hbar^2}]\tau_z\Psi+\Delta\tau_x\Psi=\mathcal{V}(x)\mathbb{I}=E\Psi$$

The finite difference scheme, which will be outlined below, is more brute force-ish, but also more tractable. We construct a grid with the x spacing denoted by $dx=\frac{L_x}{N_x}$ and the y spacing denoted by $dy=\frac{L_y}{N_y}$. For every grid point $(x_i,y_j)=(idx,jdy)$. The two component finite difference derivatives look like

$$\partial_x\Psi(i,j)=\frac{\Psi(i+1,j)-\Psi(i-1,j)}{2dx}$$

$$\partial_y\Psi(i,j)=\frac{\Psi(i,j+1)-\Psi(i,j-1}{2dy}$$

$$\partial_x^2\Psi(i,j)=\frac{\Psi(i+1,j)+\Psi(i-1,j)-2\Psi(i,j)}{2dx^2}$$

$$\partial_y^2\Psi(i,j)=\frac{\Psi(i,j+1)+\Psi(i,j-1)-2\Psi(i,j)}{2dy^2}$$

For all such (i,j) which are out of bounds, we set the wavefunction to be zero.

$$\Psi(i,j)=0|(i,j)\notin(x,y):0\leq x\leq N_x-1;0\leq y\leq N_y$$

Plugging these into the differential equation and simplifying, we obtain, yet again, a matrix eigenvalue equation

$$\frac{\hbar^2}{2m}\tau_z[\Psi(a,b)(\frac{1}{dx^2}+\frac{1}{dy^2}+\frac{e^2B^2b^2dy^2}{\hbar^2})+\Psi(a+1,b)(\frac{jeBbdy}{\hbar dx}-\frac{1}{2dx^2})+\Psi(a-1,b)(\frac{-1}{2dx^2}-\frac{jeBbdy}{\hbar dx})$$

$$-\frac{\Psi(a,b-1)+\Psi(a,b+1)}{2dy^2}]+(\Delta\tau_x+\mathcal{V}(x)\mathbb{I})\Psi(a,b)=E\Psi(a,b)$$

I introduce a few re-definitions of the terms in the equation

$$\epsilon=\frac{2m}{\hbar^2}E$$

$$\delta\tau_x = \frac{2m}{\hbar^2}\Delta\tau_x$$

$$\mathbb{V}(x) = \frac{2m}{\hbar^2}\mathcal{V}(x)$$

$$\frac{eB}{\hbar} = \beta$$

Simplifying the expression further, we see that the first term in the difference equation looks like

$$\Lambda(a,b) = [\Psi(a,b)(\frac{N_x^2}{L_x^2}+\frac{N_y^2}{L_y^2}+\beta^2\frac{b^2L_y^2}{N_y^2})+\Psi(a+1,b)(j\beta b\frac{L_yN_x}{L_xN_y}-\frac{N_x^2}{2L_x^2})+\Psi(a-1,b)(-\frac{N_y^2}{2L_y^2})$$
$$(\Psi(a,b-1)+\Psi(a,b+1))]\tau_z$$

I introduce the constants A, B, C, D as

$$A = \frac{N_x^2}{L_x^2} + \frac{N_y^2}{L_y^2} + \beta^2\frac{b^2L_y^2}{N_y^2}$$

$$B = j\beta b\frac{L_yN_x}{L_xN_y} - \frac{N_x^2}{2L_x^2}$$

$$C = -\frac{N_y^2}{2L_y^2}$$

$$D = \frac{N_y^2}{2L_y^2}$$

This implies

$$\Lambda(a,b) = \Psi(a,b)A + \Psi(a+1,b)B + \Psi(a-1,b)C + \Psi(a,b-1)D + \Psi(a,b+1)D$$

And the difference equation looks like

$$\Lambda(a,b)\tau_z + (\delta\tau_x + \mathbb{V}(x)\mathbb{I})\Psi(a,b) = \epsilon\Psi(a,b)$$

Plug in the matrix values, we see that we end with two equations, as expected for the two components ψ_+, ψ_- of Ψ.

$$(A+\mathcal{V}(adx))\psi_+ + \delta\psi_-(a,b) + B\psi_+(a+1,b) + C\psi_+(a-1,b) + D\psi_+(a,b-1) + D\psi_+(a,b+1) = \epsilon\psi_+(a,b)$$

$$(-A+\mathcal{V}(adx))\psi_-(a,b) + \delta\psi_+(a,b) - B\psi_-(a+1,b) - C\psi_-(a-1,b) - D\psi_-(a,b-1) - D\psi_-(a,b+1) = \epsilon\psi_-(a,b)$$

How do we solve this for the eigenvalues ϵ? Notice that we can treat these equations as those with $\psi_+(i,j), \psi_-(i,j)$ as the unknown eigenvector elements. That is, our eigenvector for the eigenvalue equation looks like

$$\begin{pmatrix} \psi_+(0,0) \\ \psi_-(0,0) \\ \psi_+(1,0) \\ \psi_-(1,0) \\ .. \\ .. \\ \psi_+(N_x-1,N_y) \\ \psi_-(N_x-1,N_y) \end{pmatrix}$$

And it rightly checks out with the density of states count, which is $2N_x(N_y+1)$. Once this numbering scheme is understood, all we need to do is to find out given a $\psi_{\pm}(a,b)$, what is it's vertical position in the eigenvector, if we start counting from the first element. And, this can be checked by induction that

$$\psi_+(a,b) \to 2bN_x + a + 1$$

$$\psi_-(a,b) \to 2bN_x + a + 2$$

And we are done. Now the program runs over the two equations and constructs the $2N_x(N_y+1) \times 2N_x(N_y+1)$ matrix whose eigenvalues we are working out, by plugging in the corresponding coefficients at the aforementioned position numbers. This finite difference scheme has added advantage that at the low energy spectrum it checks out with the first order approximation and hence causes no issues physically.

7 Attempts at the Finite Difference Method

This section and henceforth is my work done in the next four weeks. There are a few corrections to the earlier section. Firstly, the vertical position markers are wrong. The correct position markers for the $\psi_+(a,b)$ and $\psi_-(a,b)$ are given by

$$\psi_+(a,b) \to 2bN_x + 2a + 1$$

$$\psi_-(a,b) \to 2bN_x + 2a + 2$$

I use NumPy to do computations in Python. A simple observation lends us the fact that for a NXN sized grid, the dimensions of the matrix whose eigenvalues we are supposed to compute scale as $2NX2N$ due to the two component form of the eigenvector. Initial glitches with using Python soon appear as it seems to run out of memory for grids of edge 10^4, it gives the familiar $MemoryError$. I went on to check for smaller cases, for instance, the single grid point solution is a $2X2$ matrix and can be solved by hand. To complete this demonstration, a single grid point $(0,0)$ would yield an eigenequation which looks like

$$\begin{pmatrix} A & \delta \\ \delta & -A \end{pmatrix} \begin{pmatrix} \psi_+(0,0) \\ \psi_-(0,0) \end{pmatrix} = \epsilon \begin{pmatrix} \psi_+(0,0) \\ \psi_-(0,0) \end{pmatrix}$$

$$A\psi_+(0,0) + \delta\psi_-(0,0) = \epsilon\psi_+(0,0)$$

$$\delta\psi_+(0,0) - A\psi_-(0,0) = \epsilon\psi_-(0,0)$$

$$\implies \epsilon = \sqrt{\delta^2 + A^2}$$

An actual computation of $A(0)$ shows us that it is actually quite small compared to δ, by ten orders. If we neglect the A term, the result is quite simple

$$\epsilon = \pm\delta$$

The result printed by the Python program for the two grid point case, in which, the matrix is of dimension 4, gives us the similar result as the eigenvalues either δ or $-\delta$. Here δ is set as $0.5e-9$.

```
0  0
1  0
[[   2.44171e-24+0.j     5.00000e-10+0.j    -6.10426e-25+0.j
0.00000e+00+0.j ]
 [   5.00000e-10+0.j    -2.44171e-24+0.j     0.00000e+00+0.j
6.10426e-25+0.j ]
 [   0.00000e+00+0.j     0.00000e+00+0.j     2.44171e-24+0.j
5.00000e-10+0.j ]
 [   0.00000e+00+0.j     6.10426e-25+0.j     5.00000e-10+0.j    -2.44171e-24+0.j ]]
[(-4.999999999999998e-10+0j),
(-5.000000000000001e-10+0j),
(5e-10+0j),
(5e-10+0j)]
```

The case for higher matrix dimensions becomes difficult to compute by hand. The case with four grid points leads to an 8 X 8 matrix, if tried to solve by hand involves the solution of an eighth order polynomial equation, which when solved by a straightforward insert-into-the-program routine surprisingly gave similar results.

```
[[   9.76682e-24+0.j     5.00000e-10+0.j    -2.44171e-24+0.j
0.00000e+00+0.j
    -2.44171e-24+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j
     0.00000e+00+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j ]
 [   5.00000e-10+0.j    -9.76682e-24+0.j     0.00000e+00+0.j
2.44171e-24+0.j
     0.00000e+00+0.j     2.44171e-24+0.j     0.00000e+00+0.j
0.00000e+00+0.j
     0.00000e+00+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j ]
 [   0.00000e+00+0.j     0.00000e+00+0.j     9.76682e-24+0.j
5.00000e-10+0.j
    -2.44171e-24+0.j     0.00000e+00+0.j    -2.44171e-24+0.j
0.00000e+00+0.j
     0.00000e+00+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j ]
 [   0.00000e+00+0.j     2.44171e-24+0.j     5.00000e-10+0.j    -9.76682e-24+0.j
     0.00000e+00+0.j     2.44171e-24+0.j     0.00000e+00+0.j
2.44171e-24+0.j
     0.00000e+00+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j ]
 [   0.00000e+00+0.j     0.00000e+00+0.j    -2.44171e-24+0.j
0.00000e+00+0.j
     9.76682e-24+0.j     5.00000e-10+0.j    -2.44171e-24+0.j
0.00000e+00+0.j
    -2.44171e-24+0.j     0.00000e+00+0.j     0.00000e+00+0.j
0.00000e+00+0.j ]
```

```
 [   0.00000e+00+0.j    2.44171e-24+0.j    0.00000e+00+0.j
2.44171e-24+0.j
    5.00000e-10+0.j   -9.76682e-24+0.j    0.00000e+00+0.j
2.44171e-24+0.j
    0.00000e+00+0.j    2.44171e-24+0.j    0.00000e+00+0.j
0.00000e+00+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j   -2.44171e-24+0.j
0.00000e+00+0.j
   -2.44171e-24+0.j    0.00000e+00+0.j    9.76682e-24+0.j
5.00000e-10+0.j
   -2.44171e-24+0.j    0.00000e+00+0.j   -2.44171e-24+0.j
0.00000e+00+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
2.44171e-24+0.jX
    0.00000e+00+0.j    2.44171e-24+0.j    5.00000e-10+0.j   -9.76682e-24+0.j
    0.00000e+00+0.j    2.44171e-24+0.j    0.00000e+00+0.j
2.44171e-24+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
0.00000e+00+0.j
   -2.44171e-24+0.j    0.00000e+00+0.j   -2.44171e-24+0.j
0.00000e+00+0.j
    9.76682e-24+0.j    5.00000e-10+0.j   -2.44171e-24+0.j
0.00000e+00+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
0.00000e+00+0.j
    0.00000e+00+0.j    2.44171e-24+0.j    0.00000e+00+0.j
2.44171e-24+0.j
    5.00000e-10+0.j   -9.76682e-24+0.j    0.00000e+00+0.j
2.44171e-24+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
0.00000e+00+0.j
    0.00000e+00+0.j    0.00000e+00+0.j   -2.44171e-24+0.j
0.00000e+00+0.j
   -2.44171e-24+0.j    0.00000e+00+0.j    9.76682e-24+0.j
5.00000e-10+0.j]
 [   0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
0.00000e+00+0.j
    0.00000e+00+0.j    0.00000e+00+0.j    0.00000e+00+0.j
2.44171e-24+0.j
    0.00000e+00+0.j    2.44171e-24+0.j    5.00000e-10+0.j   -9.76682e-24+0.j]]
[(5.000000000000001e-10+0j),
(-5.000000000000003e-10+0j),
(-4.999999999999991e-10+0j),
(4.999999999999994e-10+0j),
(4.999999999999999e-10+0j),
(-5e-10+0j),
(-5.000000000000001e-10+0j),
(5.000000000000002e-10+0j),
(-4.999999999999999e-10+0j),
(5.000000000000002e-10+0j),
```

(−5.000000000000001e−10+0j),
(4.999999999999999e−10+0j)]

If noted in the last two data files, the main matrix is mostly zeros. These special matrices have a name, they are called **sparse matrices**. Sparse matrices have the added advantage that they have much quicker algorithms for eigenvalue computations. There are another special set of matrices called **banded matrices** in which the only non zero elements in the matrix lie around the diagonal of the matrix. What is special about the matrices we are dealing with is that, each main matrix is sparse, and has a **block matrix** form. By a block matrix form, we mean, we have a matrix whose entries can be classified into submatrices which repeat. This is also apparent from the specimen matrices shown in the data files.

The matrices we deal with here are banded in their block matrix form. This observation, which struck me when I was solving the 4 grid point case by hand and was trying to get the matrix form, is of good use when trying to speed up the execution of the program to many orders. For the sake of discussion, let us write the four grid matrix directly.

$$\begin{pmatrix} A & \delta & B & 0 & 0 & 0 & 0 & 0 \\ \delta & -A & 0 & -B & 0 & -D & 0 & 0 \\ D & 0 & 0 & 0 & A & \delta & B & 0 \\ 0 & -D & 0 & 0 & \delta & -A & 0 & -B \\ B^* & 0 & A & \delta & 0 & 0 & D & 0 \\ 0 & -B^* & \delta & -A & 0 & 0 & 0 & -D \\ 0 & 0 & D & 0 & B^* & 0 & A & \delta \\ 0 & 0 & 0 & -D & 0 & -B^* & -\delta & -A \end{pmatrix}$$

Grouping the repeating submatrix elements and rewriting the 8X8 matrix into a four by four matrix with two by two submatrix elements. The new matrix representation now looks like

$$\begin{pmatrix} A & B & D & 0 \\ D & 0 & A & B \\ B^* & A & 0 & D \\ 0 & D & B^* & A \end{pmatrix}$$

It is visibly clear that the transpose conjugate or the Hermitian conjugate of the matrix is itself, thereby immediately implying that whatever eigenvalues it gives are purely real as they should be considering that we are dealing with energy eigenvalues. The block form of this matrix makes an attractive choice for statistical programming languages like R to do relatively quicker computations. I could use this feature by coupling R with my Python program.

I would also like to discuss a few pathological issues with NumPy as a tool.

One of the main issues while working with small numbers and physical situations like this is that NumPy initializes matrices differently for different function calls. The program given below is the final version of the finite difference program for the four point case.

```
MAIN = np.empty((DIM,DIM),'complex')
for b in range(Ny+1):
    for a in range(Nx):
       print(b,a)
       if(0 <= (2*Nx*b + 2*a) <= DIM):
          MAIN[2*Nx*b + 2*a, 2*Nx*b + 2*a] =
```

Notice the instantiation of the matrix as an empty matrix. The output gives non zero, but extremely small real and imaginary parts in the matrix. These imaginary parts, are, that too, instantiated at random. In this case I set the magnetic field to zero and hence, intrinsically all imaginary terms are supposed to be zero. For a non zero magnetic field, say 1 T, the imaginary values get fudged.

```
 2.12200e-312 +4.94066e-324j   0.00000e+000 +0.00000e+000j
 6.10426e-024 +0.00000e+000j   5.00000e-010 +0.00000e+000j
-2.44171e-024 +0.00000e+000j   5.30681e-316 +0.00000e+000j
-6.10426e-025 +0.00000e+000j   0.00000e+000 +1.03754e-322j]
 2.12200e-312 +9.88131e-324j   2.44171e-024 +0.00000e+000j
 5.00000e-010 +0.00000e+000j  -6.10426e-024 +0.00000e+000j
```

If I now instantiate the matrix as a matrix of complex valued zeros,

```
MAIN = np.zeros((DIM,DIM),'complex')
for b in range(Ny+1):
    for a in range(Nx):
       print(b,a)
       if(0 <= (2*Nx*b + 2*a) <= DIM):
          MAIN[2*Nx*b + 2*a, 2*Nx*b + 2*a] = fA(b)
```

The matrix entries are perfectly real for the values we are working with. Setting a non zero magnetic field, the values turn out perfectly right.

```
 5.00000e-10 +0.00000e+00j
 0.00000e+00 +0.00000e+00j]
 6.10426e-25 +0.00000e+00j
 2.44171e-24 +1.85480e-23j
-1.47001e-22 +0.00000e+00j
 2.44171e-24 -1.85480e-23j]
 0.00000e+00 +0.00000e+00j
 0.00000e+00 +0.00000e+00j
 0.00000e+00 +0.00000e+00j
```

This apparent differences in instantiation led to many initial faults in the simulation. The garbage imaginary values picked up in the former case are picked up from the memory when it discards the imaginary component at some step and tries to retrieve for computations at a later step.

The next idea is to use R in conjunction with Python. The issue here is again purely based on the limitations of the language under use. R numbers its entries from 1 onwards. That is the matrix elements are not labeled as $(0,0),(0,1)$ et cetera, but are labeled as $(1,1),(1,2)$. As small as the issue looks, it needs a bit of patchwork. The vertical indexing of the eigenvector has to be slightly modified to fit the scheme in R. Again, as easy as it seems, it leads to further

complication when assigning matrix entries their right values. But this is actually useless as we do not need R for constructing the matrix, rather, we need R to solve for the eigenvalues quickly because Python is slow while extracting eigenvalues but is decently fast while constructing the matrix. The entire procedure is done in three steps. A Python script to write the matrix to a file. A Python script to binarize the matrix data which is then piped into an R program which compiles the binary data to run the appropriate eigenvalue solver for computation. The advantage with R is that the selection of the algorithm is mostly automated.

```
#!/usr/bin/r

infile <- "C:/Users/Anurag Pallaprolu/Desktop/IISc/hello.bin"
con <- file(infile, "rb")
dim <- readBin(con, "integer", 2)
Mat <- matrix( readBin(con, "numeric", prod(dim)), dim[1], dim[2])
close(con)

print(Mat)

eig(Mat, show_values = YES)
```

8 Further Reading on Eigenvalue Algorithms

The central idea of computing eigenvalues for large matrices is to reduce the number of computations by classifying the matrices to certain known forms, we have already seen the sparse and the block matrix forms. In regards to this, I mention two theorems which lay foundation to any eigenvalue algorithm.

Schur Decomposition If A is a N dimensional matrix, then there exists B such that we can construct a upper triangular C with the relationship

$$B^*AB = C$$

The diagonal elements of C are the eigenvalues desired.

Schur Decomposition for Real Matrices If A is a real N dimensional matrix, then there exists an orthogonal real B such that we can construct an upper triangular C such that

$$B^TAB = C$$

The diagonal elements of C are either $1X1$ or $2X2$ blocks. If they are of the latter, then there is a complex conjugate eigenpair.

Power Iteration Most of the commonly used eigenvalue finding algorithms are iterative and in most cases one does not end up with computing all the eigenvalues. The **power iteration** method, which is used by Google in its PageRank algorithm, is a simple procedure in which one ends up computing the dominant eigenvector in the spectrum. One starts off with a test vector v_0 and

calculates the new updated vector

$$v_{k+1} = \frac{Av_k}{|Av_k|}$$

After a few iterations, it is noted that the vector v_{inf} converges to the eigenvector corresponding to the largest eigenvalue. The proof of this procedure is given below and is quite simple. Let us write v_0 in the eigenbasis of A.

$$v_0 = \sum_k a_k u_k$$

where u_k are the eigenbasis. Now, after some mth iteration, we see that

$$v_m = A^m v_0$$

$$\implies v_m = A^m \sum_k a_k u_k$$

$$\implies v_m = \sum_k a_k A^m u_k$$

$$\implies v_m = \sum_k a_k \lambda_k^m u_k$$

Let us assume that for some j, we have $|\lambda_j| \geq |\lambda_i|$ for all i.

$$\implies v_m = \lambda_j^m a_j u_j + \sum_{i \neq j} a_i \lambda_i^m u_i$$

$$\implies v_m = \lambda_j^m (a_j u_j + \sum_{i \neq j} a_i \frac{\lambda_i^m}{\lambda_j^m} u_i)$$

Tending m to infinite, we see that the summation term tends to zero as you are exponentiating a quotient less than one. This then immediately proceeds to the conclusion.

Lanczos Iteration This technique uses the previous iteration method to compute eigenvalues. It is used in FreeFEM++ as a standard procedure to solve eigenvalue PDE problems. The vectors which we were working with in the previous method, v_j all belong to a subspace of the vector space spanned by the eigenbasis called the **Krylov** subspace. The Lanczos method deals with these vectors and tries to iteratively generate vectors tending to eigenvectors of the matrix by orthogonalizing and then processing them.

The Algorithm Generate a random unimodular vector u. Now let $v_0 = 0$ and $v_1 = u$. Also call $w_1 = 0$.

$w_j = Av_j$

$\gamma_j = w_j.v_j$

$w_j = w_j - \gamma_j v_j - \beta_j v_{j-1}$

$\delta_{j+1} = |w_j|$

$v_{j+1} = \frac{w_j}{\delta_{j+1}}$

Iterate the steps above for j running from 1 to one less than the dimension of the matrix. For the nth entry, we see that

$$w_n = Av_n$$

$$\gamma_n = w_n.v_n$$

Now there is a mathematical theorem which proves that the eigenvalues of A will be the same as that of the triply banded matrix below. Now that the matrix is banded, we can compute the eigenvalues in a worst case of quadratic time using the QR algorithm or a modification of the power iteration procedure.

$$\begin{pmatrix} \gamma_1 & \delta_2 & 0 & 0 & 0\dots & 0 \\ \delta_2 & \gamma_2 & \delta_3 & 0 & \dots & 0 \\ 0 & \delta_3 & \gamma_3 & \delta_4 & \dots & 0 \\ & & & & & \\ & & & & & \\ & & & & & \\ 0 & \dots & \dots & \dots & \delta_n & \gamma_n \end{pmatrix}$$

9 Particles, Gravity and Strings Conference. Christ University, Bengaluru

This section/digression is a review of the Particles, Gravity and Strings conference held at the Christ University in Bengaluru on the 27th of June, 2015. The event also marked a century of relativistic physics, marking the discovery of the laws of General Relativity by Einstein in 1915. There were two sessions, the morning session consisting of short and informative talks by Nima Arkami-Hamed from UC, Berkeley and Ashoke Sen from Harish Chandra Research Institute, and the afternoon session consisting of public lectures delivered by Nathan Seiberg, Andrew Strominger and Cumrun Vafa. Due to lack of time, I was not able to listen to Professor Vafa's lecture, but the rest of the program has been documented.

N.Arkami-Hamed started off his talk with the major triumph in twentieth century physics, that is Quantum Field Theory. The unification of Relativity and Quantum Mechanics leading to a coherent theory which explains the universe to almost everything(except gravity, not surprisingly). His next question was on a similar tone, why do we have a macroscopic universe? He says that we can answer this question to sufficient accuracy but today's science has a rather lousy explanation. He then mentioned a fact which was a pretty obvious but non trivial, and is something most people studying HEP or QFT would overlook. All fundamental interactions end up using single vertex Feynman diagram representations. For instance the gravitational, electromagnetic, strong and weak forces, which all admit a vector boson mediator have a single vertex three particle structure. N.Arkami-Hamed asks why there can't be a *multiparticle* single vertex diagram. For instance, in the electromagnetic case you define a Coulomb

potential as a potential between *two* charged particles and then go on to define the interaction potential of a charge distribution based on this two charge definition. Another fundamental question which was raised was that in regards to the magnitude of fundamental spin that can be assigned to a particle. The maximum allowed spin for a Standard Model particle, the graviton, is $2\hbar$. Why can't there be a particle with a fundamental spin $7\hbar$?

The recent hype created by the discovery of the Higgs boson led to the discussion on how the Higgs mechanism proceeds. The vector bosonic matter in the world can be classified as either a massive or a massless vector boson. One of the many differences, apart from the obvious is that massive vector bosons have three degrees of freedom while the massless ones have only two, better known by the term "Helicity". The Higgs creates the mass by adding the extra degree of freedom. The wonderful fact about the Higgs is that it is the first particle which has been theoretically hypothesized and then discovered, a very important driving force for the correctness of the Standard Model. He then quips about the fact that the particle of spin $\frac{3}{2}$, or the Rarita-Schwinger particle hasn't been discovered yet, and strongly believes in the fact that Supersymmetry or SUSY might be the way to its discovery.

Ashoke Sen then took the stage and started his talk on String Theory. At its essential core, the theory is an attempt at a Grand Unified Theory with the corresponding fundamental entities being strings rather than particles. Strings, which are supposed to be existant at a length scale of 10^{-33}. The theory has a one dimensional fundamental entity as compared to the Standard Model which handles particles, intrinsically zero dimensional. The entire attempt at a theory different from the Standard Model was because the absence of gravity being accounted as a QFT leads to the question, what is the outcome of Gravity + Quantum Mechanics? There have been many attempts at trying to answer the question(Loop Quantum Gravity being one) but String Theory looks like a long running candidate. Again, only precise experimental verification would attest to the correctness or the failure of a scientific theory. Mathematicians have apparently worked out the fact that if there ever existed a String Theory, then the theory would be unique, unlike the Standard Model, which is just another one of the infinitude of quantum field theories that exist.

Sen then went on to explain the theory on a very basic level. Just as in classical mechanics, where a particle in a hilly potential ends up getting trapped in one of the wells, in string theory too, our Universe is expected to be trapped in one of these wells, which are technically known as landscapes. The issue is there are many landscapes and these aren't neccesarily three dimensional, even ten dimensional ones exist. The job of string theory is to find the landscape which explains our three dimensional world correctly, and by that we even include gravity into the picture. He finally approached the question of how string theory could be tested. A direct conclusive evidence of the existence of strings would be to observe a string in a supercollider experiment. The Large Hadron Collider can resolve particles of dimensions of the order 10^{-17} and to resolve further we need to collide particles with higher and higher energies.

However, indirect tests can be performed. For instance, it is well known from

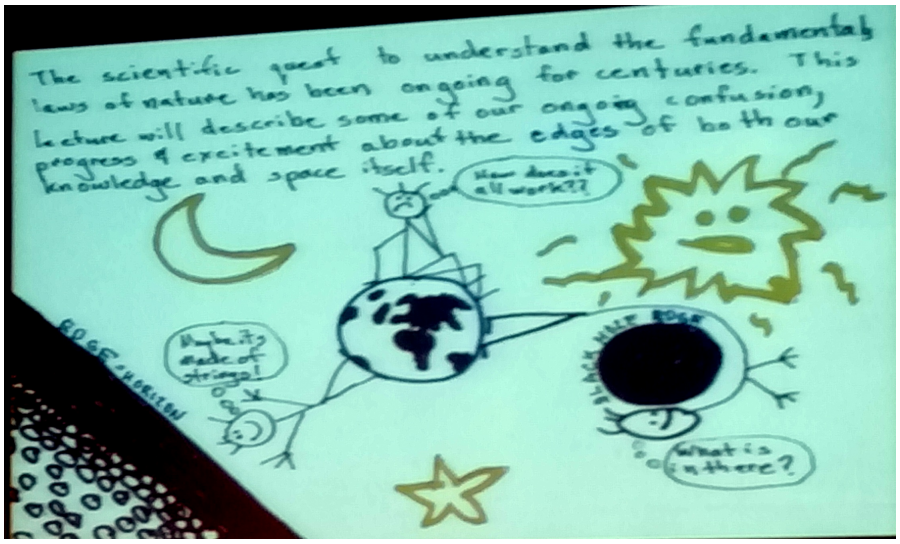
particle physics experiments that the time of decay of a particle is inversely proportional to the strength and the range of the forces binding it. As an example, if you place a neutron on the table, it would disappear into an electron, a proton and an antineutrino. This happens in 15 minutes. On the other hand, subnuclear particle decays happen at orders much higher than the neutron decay. The explanation for this is simple, the neutron is bound by weak forces and the subnuclear particles are bound by strong nuclear forces. Recent interest has come up in the question of proton decay. Protons are supposed to be very stable inside the nucleus but have been predicted to have decay times of the order of 10^{41} seconds. This time scales with the age of the universe, which literally means all the protons created from the beginning of time have almost all survived. Now, if there ever was a proton decay observed, then the decay field must be weaker than any fundamental force observed. This could be an indirect test of string theory. The study of Cosmic Microwave Background Radiation could also lead to new insights in the verification of the theory.

The second phase of the programme involved three talks delivered by Nathan Seiberg, Andrew Strominger and Cumrun Vafa. The present section involves the first two talks only.

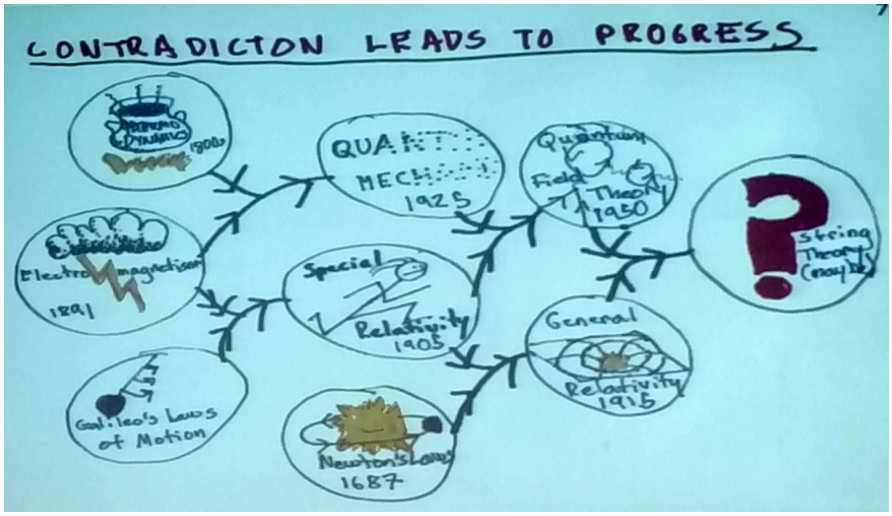
Seiberg's talk was titled "The Frontiers Of Fundamental Physics". With the discovery of the Standard Models for both elementary particles and cosmology, it might have given the image that we have achieved whatever we could in Physics. However, he cites the document "Ninteenth Century Clouds over the Dynamical Theory of Heat and Light" by William Thomson, who is better known by the name Lord Kelvin. The paper, published at the turn of the century, involves the discussion of the two major discrepancies in Physics of that time, the issues of Galiliean transformations not reproducing the same electrodynamic physics and the mismatch in the experimental specific heat values with those theoretically computed, along with that of blackbody radiation. He called these issues "clouds" for there was no explanation for them at the time. But resolution of each of these issues led to two major fields of Physics, those of special relativity and quantum mechanics. He then states that, similarly, the Physics we are at now does have some clouds and they might lead to some revolutionary ideas in both mathematics and physical sciences.

In a way very similar to how Mendeleyev predicted the properties of unknown elements with the help of the elements surrounding it, Seiberg argues that there should be a similar explanation to the way the elementary particles are arranged in the Standard Model. Questions such as those regarding the existence of non zero neutrino masses, the correctness of loop quantum gravity, and many more can be explained if one can decode this pattern in the standard model, he stated. He then went on to address the issues with cosmology and about how the two major "clouds" of cosmology, Dark Matter and Dark Energy, are yet to be explained by a coherent theory. The fact that current day physics would not be valid once one goes beyond Planck dimensions($L_p = 10^{-35}m$ and $\tau_p = 10^{-43}s$) was exposited with string theory being the strong contender to explain pre-Planck scale physics. The concept of Multiverses, each universe with its own cosmological constant was explained, with the startling idea that our universe with its cosmolgical constant lead to life on Earth acknowledged.

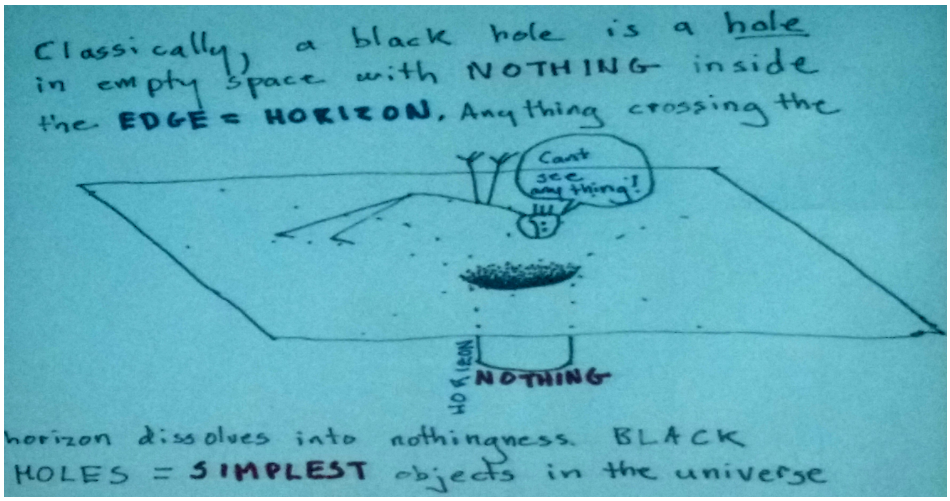
Andy Strominger's talk was titled "The Edge of The Universe: Black Holes, Horizons and Strings" and was, contrary to the other talks, entertaining due to his magnificient ability to sketch his slides with comical perfection and his uncanny way to explain deep facts in advanced physics. I shall include a few slides below for demonstration purposes, photographed at the event, to end this section.



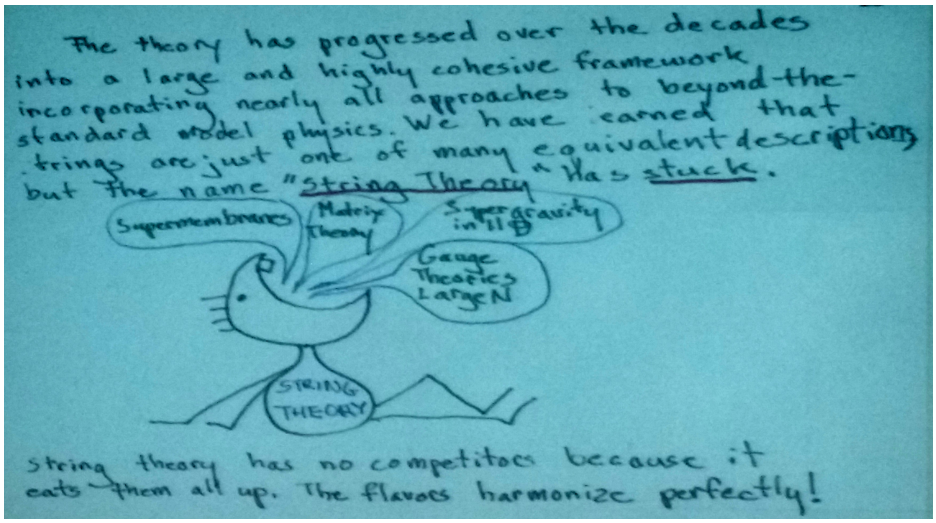
The introductory slide



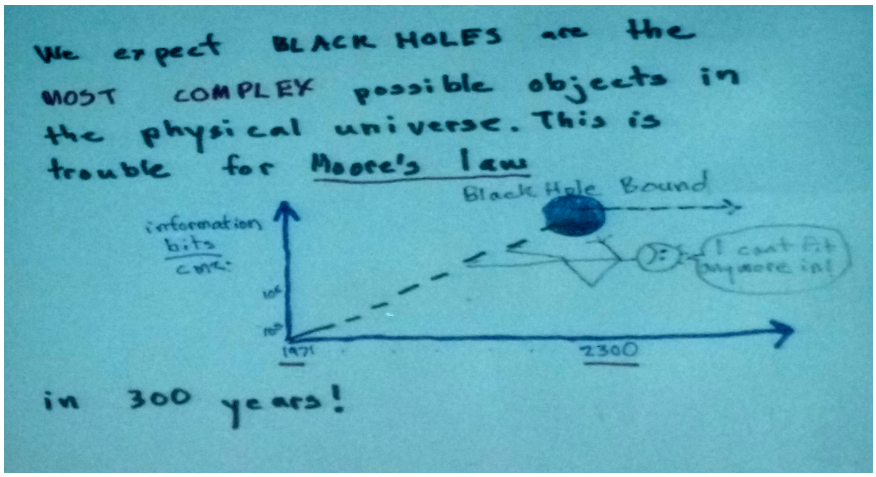
"Contradiction Leads To Progress"



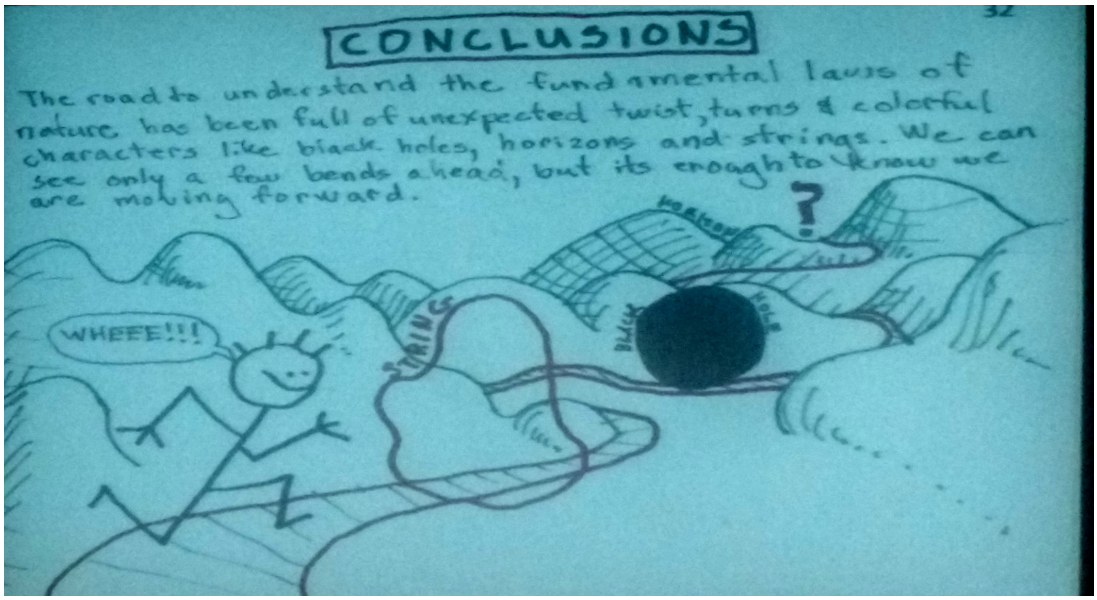
What is a black hole?



What is string theory?



Moore's Law and Black Holes



Conclusion