Introduction:

It was our goal to create a program that would model electron transport in materials and tell us something about the electron band structure. There are various methods in which physicists use in order to do this. But for the scale of bulk materials, the robust power and versatility of the tight binding model is ideal.

The tight binding model is a tool used to quantitatively describe the electronic structure of solids. It is the opposite extreme to a nearly-free electron model in that the binding potential is so large that the electrons are confined closely to the lattice sites. In a first approximation of the system, only hopping to adjacent sites is possible. Bloch wavefunctions can be used to construct orbitals in the unit cell which may then be repeated throughout space to construct the crystal lattice, hence a description of electron motion is given. These are some of the basic ideas that may then be used to describe the electron transport of any crystal.

For a 1D chain of atoms the Hamiltonian will take the form of:

$$H\psi k = E\psi k,$$

$$\langle \phi n|H|\psi k \rangle = E\langle \phi n|\psi k\rangle$$

The dispersion relation may then be written as:

$$E = \epsilon - 2t*\cos(ka)$$

With ϵ as the on-site energy and t is the barrier energy. The k is related to the electron momentum.

However, we decided to start off with the most simple energy equation, as we wanted to get a temperature dependent electron movement simulator up and running before introducing more advanced phenomena such as variable electron momentum. As such, by referring to our Solid State Physics textbook and online literature, we decided to start by using the energy value:

$$E(\text{stay}) = \epsilon \, , \, E(\text{hop}) = \epsilon + J,$$

Where J is the hopping strength (approx. 2.6 eV).

There are various assumptions and simplifications that we made in order to make our model more computationally feasible:

- All atom sites are identical
- One free valence electron/hole per atom (can be scaled up to accommodate # of orbitals)
- Energetically unfavorable for two electrons to occupy this valence site so we made it impossible (simplest case)
- Nearest neighbor hopping only
- Equal probability to hop to adjacent unoccupied sites

Code Description:

- Electron_placement:

  Was used to give the electrons a random initial distribution over a specified space. However, an important stipulation is that these starting points must lie on a point of a hexagonal lattice so that when their positions are input into the motions function they will trace out the hexagonal pattern that is characteristic of graphene. The array named

"electronarray" is what will collect all of the electron's positions and is ultimately what will be passed around to each of the subsequent functions. Based on the conditions of the tightbinding model, specifically that we have forbidden the overlap of electrons, we needed to make sure that none of the electrons were positioned on top of one another. Though, without the overlap check the probability and motion functions would have rectified the problem in most cases. However, the possibility of two electrons being stacked and all three neighboring sites being occupied could lead to an infinite loop. The same would result from any time there were 5 or more electrons on a site and would be possible based upon order of change anytime there is a stack up on one site and a total of 5 electrons shared between that site and its three neighbors.

- electron_overlap_check:

  Is fed an electron's hypothesized new array (containing its x, y, and jtype) along with its current array with those values, along with which electron it is in the array (labeled i). It then checks the values of the new array with the entirety of electronarray, using an "equivalent" statement to make sure there will be no overlap. In the event that there is overlap, the current x and y position are fed into the motions() function in order to register the electrons movement for that time step.

- Probability function (prob()):

  Evaluates the nearest neighbor positions based off of the input jtype, and checks the electronarray to see if any of those positions are occupied. If so, the Boltzmann probabilities are changed accordingly. After they have been calculated, the probabilities associated with hopping to an unoccupied site and staying are calculated, returning an array with the hopping probability associated with each site as well as the probability of staying.

- Motions():

  A random number between 0 and 1 is generated, and the location, jtype and temperature fed into prob() to get the respective probability values for each outcome. The sum of the probabilities is used (with the probability for hopping to occupied orbitals set to zero) to scale the random number. Afterwards, the random number is fed into the series of probabilities in ascending order using if statements (if less than the probability, execute the action. If greater, move on to the next probability). With the occupied orbitals having probability zero, that probability is automatically skipped over. When the correct action is reached, the new x and y values are declared, the jtype changed if hopping occurred, and these three values are returned.
- Call_function(): Is fed the electronarray that is the output of the initialization function, as well as the number of electrons, time steps and temperature. There is a loop made through all the electrons for each time step, with the coordinates and jtype of each electron at that moment (condensed into the electronarray that is updated throughout) fed into the motions() function to simulate its movement. The end array for each time step is fed into the "electrons" array. The jtype count for each electron at the last time step is then averaged and returned for certain plots. At this point, a loop is run to plot each electrons movements throughout each time step.

- Temperature sweeps:
  - Hopping probability: At each temperature value, the average j value from the call_function is calculated, and divided by the number of time steps to get the fraction of the time that an electron hops for a given time step. This is then plotted as a function of temperature.
  - Carrier_Density(): The temperature is an input, and an equation with the values from some constants are defined, with the calculated carrier density value at the output

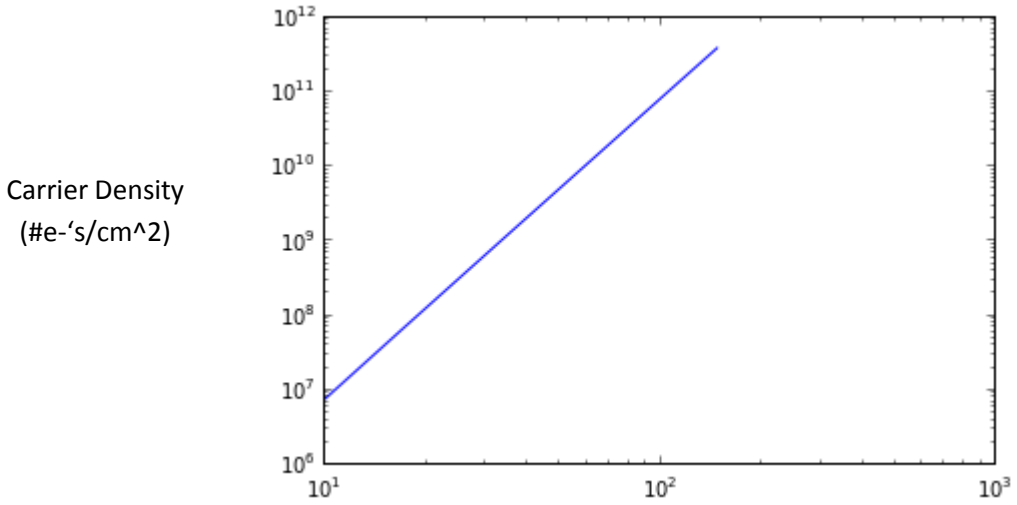
- Tight Binding Simulation: We use a base_N value calculated from looking at research literature online, and multiply it with the hopping probability associated with the respective temperature values for a given sweep.

Analytic Solution:

Our model has characteristics which might lend itself to an approximation of the carrier densities for different temperatures in graphene. We found from a research paper online the function describing the carrier concentration versus temperature and plotted the expected results. This graph closely matched the measurements of carrier concentration in graphene sheets presented in a paper published by Applied Physics Letters:

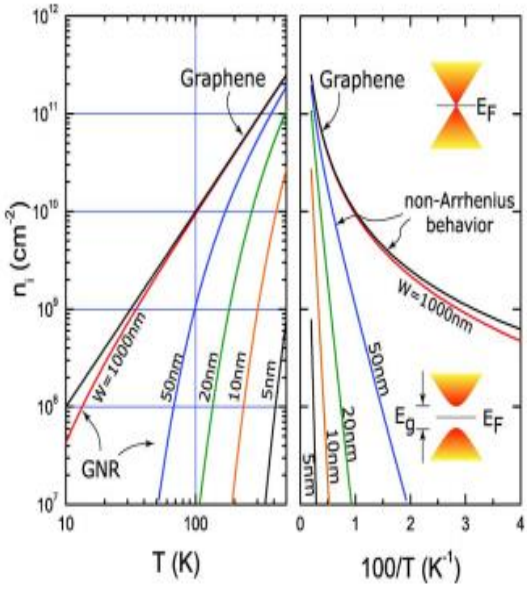
(http://www3.nd.edu/~mind/research/MINDpubs/07_6CarrierStatisticsQuantumCapacitanceGraphene.pdf).

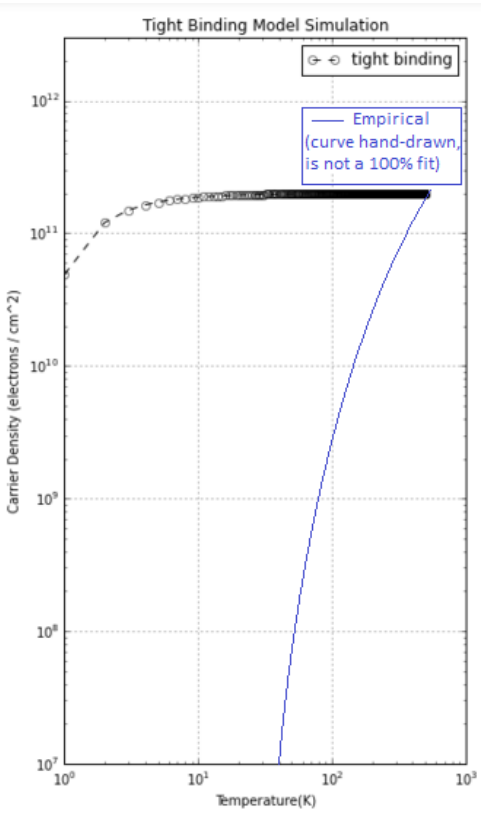
These results were off by only a factor a 10 at the two end points of the graph from their measurements. As such, this seemed to be a good approximation, given the sensitivity of the function, and so it was used as a test of the accuracy of our model.



Test Case:

The results of our test case were somewhat interesting to see. We have some working ideas about how our results miss the mark. Aside from the assumptions and simplifications that are inherent to the tight binding model, there are numerous sources of error. Some of these might include lack of computing power, forcing us to make simplifications. This comes into play for the carrier density function that results in 10e10 to ~10e30 electrons, depending upon the temperature. This forced us to find some other means of calculating it. First we thought to use a set number of electrons to evolve through the time steps and then multiply our resulting average hopping probability from the electrons by the calculated carrier density. However, you will notice that that approach doesn't show us anything except for the behavior of our probability function were this quantity to be plotted against temperature. Next, we tried to fit our data to the graph:





To the top right you will see our graph comparing our solution from our tight-binding model to a curve representing the empirical solution for a 50nm wide graphene nanoribbon (based off of a graph in the same research paper in the link provided previously). We originally tried to model a graphene sheet, but we realized that our results were far off due to exotic phenomena specific to graphene sheets (such as band gap overlap) that our code does not accommodate for (and would be quite difficult to implement). As such, our work is a better emulator of graphene nanoribbons, which was indeed an unexpected result. With our tight binding model being based essentially solely off of the Boltzmann distribution and the situation in reality being quite complicated, our model fell a bit off the mark due to it not being very well equipped to describe this complex phenomena.

Description of full physical system:

It was our hope that with the use of this model for electron transport, we could possibly model real materials. For this, the 2 dimensional lattice of graphene was chosen. Graphene consists of a hexagonal array of carbon atoms on a single plane (in the monolayer case, and even this is an approximation due to some 'scrunching up' of the lattice). Our model was then to show the transport of electrons as we evolve the system through time steps. This transport, later on in our project after establishing temperature dependence, was to use the dispersion relation for graphene which is as follows:

$$E = \pm\sqrt{\gamma_0^2\left(1 + 4\cos^2\frac{k_y a}{2} + 4\cos\frac{k_y a}{2}\cdot\cos\frac{k_x\sqrt{3}a}{2}\right)}$$

It was then our plan to find a means of describing the probability of whether the electrons were going to move to neighboring orbitals. This function must be temperature dependent. Incorporating temperature would then allow us to find carrier concentration in the lattice.

In the simple case, we consider there to be only one orbital per atom site in a hexagonal lattice. This is justified by the fact that:

> "Carbon atoms have a total of 6 electrons; 2 in the inner shell and 4 in the outer shell. The 4 outer shell electrons in an individual carbon atom are available for chemical bonding, but in graphene, each atom is connected to 3 other carbon atoms on the two dimensional plane, leaving 1 electron freely available" (http://www.graphenea.com/pages/graphene-properties)

Discussion

- Interesting: We found interesting that we were able to clearly see the temperature dependence of the electron mobility based off of the longer paths the electrons took for higher temperatures in the attractive graphs using our call_function. Being able to easily plot out the Boltzmann Distribution by plotting the hopping probability with temperature was also an interesting manifest of our model, having the distribution provide a clear physical significance in our model.
- Surprising: We found surprising, at least momentarily, how our results mismatched the analytical results. Upon further research and quickly seeing how graphene's exotic characteristics strongly affect its carrier density with temperature, we realized that this was not so surprising after all. Seeing that this also led to our model better matching the

behavior of graphene nanoribbons, which do have a band gap was a very surprising turn of events.

- Expected: In terms of things that we were not so surprised by, one was that our final plot strongly resembles the Boltzmann distribution we based our model off of. Another expected occurrence was that, when we introduced k distributions into our energy terms, that the resulting spikes from either positive or negative values did not add much of a difference into our results, considering that they were all manifesting as a series of cosine terms.

Results:

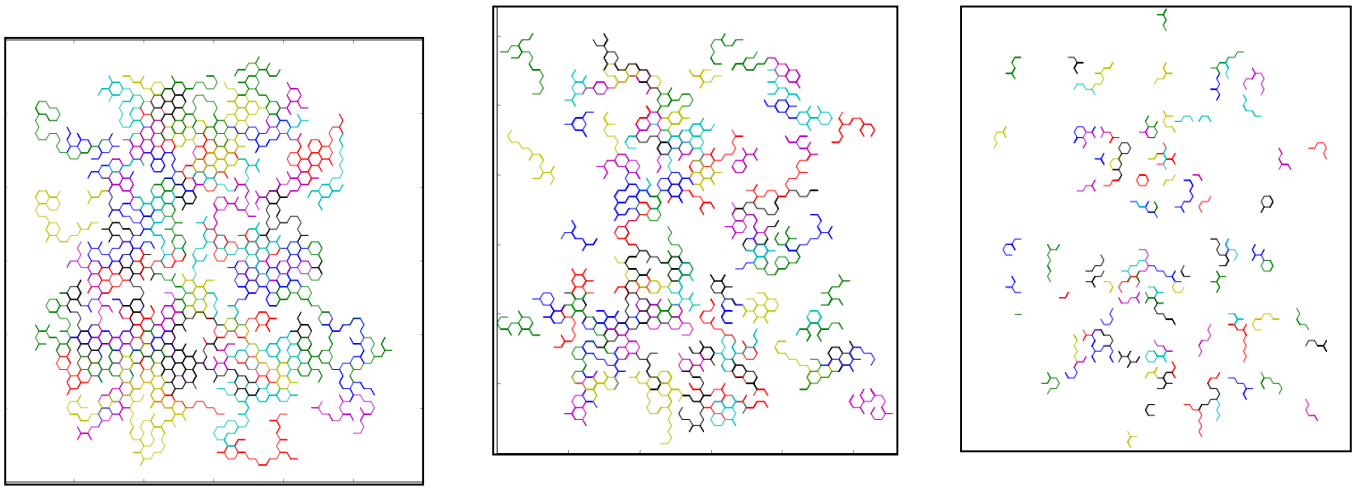
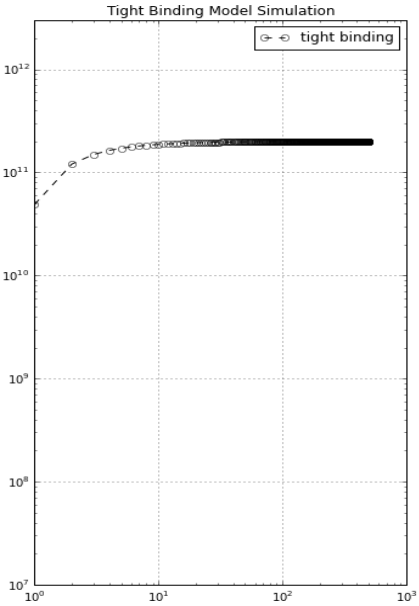
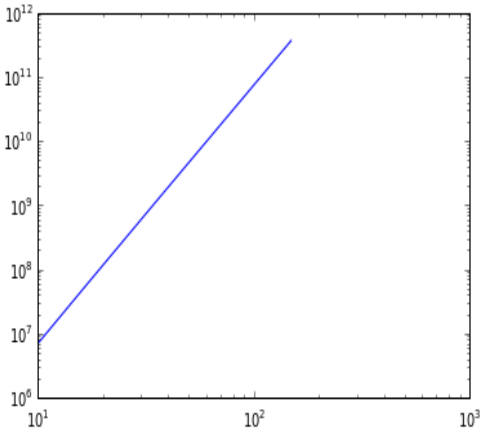
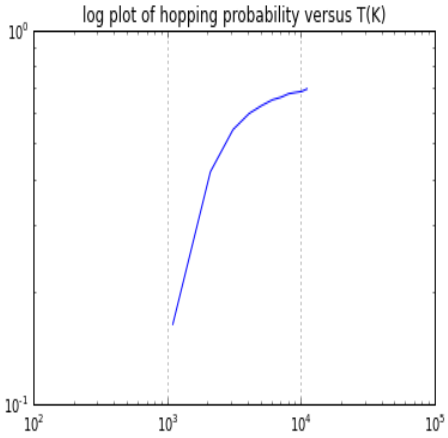


Fig: The temperature dependence on electron motion. Paths of electrons at 900K (right), 1500 K (center), 3000 K (left).







Top Left: Log-log plot of our carrier density with temperature, using a base electron number and Boltzmann Distribution. Top: Log-log plot of graphene's carrier density with temperature, using equations found on a research paper. Top Right: Log-log plot of ourhopping probability, with Temperature.

Future Direction:

If we were to have more time to develop this code, there are numerous ways in which it could be improved. The most obvious effort we would make would be to sit down and try to get a better understanding of the theory involved with carrier movement in the complex material that is graphene, as well as think about how we could extend our current model to accommodate for the various phenomena at play. That being said, we would also have to look back on what we already did and think about how we could capture these other phenomena with our model while simultaneously setting up our code in an efficient way, so we can compute these quantities and

behaviors in a realistic time scale. Previously mentioned, the tight binding model has great versatility which depends on what time of approximations you wish to make. If we were to undo some of our simplifications in an attempt to get more accurate results, the first would be the number of orbitals per atom site. This could allow electron overlap, but there would be an energy penalty that would show up in the probability function. Another would be us trying to provide a more rigorous treatment of the involvement of the electron momentum in atomic hopping, as we feel that this may be the biggest hurdle, but also the biggest key in creating a more accurate and successful test case. In this vein, we would have liked to find in relevant literature a probability that accurately depicts the chances of atomic hopping, as well as the effect of temperature on the electron's average momentum in graphene. We ran into trouble using the Fermi function, so we went with the Boltzmann probability function. We didn't get to make an accurate quantitative comparison of carrier concentration, so attempting the Fermi function could prove to be of great importance as a route to take in increasing the accuracy of our model. If we were able to get our test case on track, another continuation of our code could be to increase functionality in portraying the behavior at play, such as creating band structure graphs using the dispersion relation. This would have been as simple as plotting the energy values associated with various values of kx and ky. Finally, our code could be used to represent other lattice structures by changing current functions, like our "motions" function in order to represent movement to adjacent lattice sites such as in a rectangular lattice, as well as accommodating for the changing in hopping probability, depending on what atoms the electron has as its nearest neighbor at a given time step. Overall, although the lack of a very accurate test case was quite discouraging, this project was very much a learning experience in learning how to better collaborate on code and how to make general simulations along with general-purpose code, as well as later highly

altering that code for computational efficiency or higher accuracy, requiring an in-depth knowledge of the workings of the code and how our various functions were fed into each other and generally interrelated.

Contributions:

Both:

- Made changes and contributions to most of the functions that are in our current program
- Put together the power point and presentation

Alex:

- Developed the initialization, motion, and “call_function” functions
- Created the probability function that superseded the overlap check function
- Simulated temperature dependence of electron hopping (using call_function)
- Used final versions of functions to complete test case via implementing temperature sweeps of hopping probability and carrier density(along with stripping down the functions in order to make the computations take a reasonable amount of time)
- Wrote the Code Description (all the code except for electron_placement), the Discussion, Results, Test Case, and describing the Results, as well as making revisions and additions for the Final Paper.

Giordano:

- Rewrote the call_function, motion and electron placement functions to make them more efficient(through vectorization) and work together

- Wrote the overlap check function to work recursively with the motions function to avoid overlap, as well as updated initialization to prevent initial overlap
- Wrote the carrier density function
- Wrote a good portion of the first draft of our final paper, such as the Introduction, Physical System Description, the electron_placement Code Description, Analytic Solution and Future Direction

Code:

Can be found in the “Untitled.ipynb” section of our repository. We enjoyed your class and hope you enjoy your break!

References:

http://www3.nd.edu/~mind/research/MINDpubs/07_6CarrierStatisticsQuantumCapacitanceGraphene.pdf

http://www.graphenea.com/pages/graphene-properties

https://en.wikipedia.org/wiki/Boltzmann_distribution

http://lampx.tugraz.at/~hadley/ss1/bands/tightbinding/tightbinding.phpki/Tight_binding

http://lampx.tugraz.at/~hadley/ss1/bands/tightbinding/tightbinding.php