

Numerical Calculation of Density of States and Band Structure

A Thesis Submitted in Partial Satisfaction
Of the Requirements for the Degree of
Bachelor of Science in Physics
at the
University of California, Santa Cruz

By
Chao Wang
Jun 10, 2009

Keivan Esfarjani
Technical Advisor

David P. Belanger
Supervisor of Senior Theses, 2005-2009

David P. Belanger
Chair, Department of Physics

ABSTRACT

Objective: write a program in Mathematica to calculate Density of States (DOS) and Band Structure (BS) of a crystal structure when given lattice constants and periodic potential in real space. This program provides a concise and simple method of calculating DOS and BS, and therefore provides a basis for numerical optimization of various properties of a nanostructured system. The program is tested under known conditions and the results closely match the known result.

1. INTRODUCTION

Energy consumption has become an important issue lately, with global shortage on commonly used fossil fuel energy sources such as oil, natural gas and coal. Unfortunately, a very large percentage of all energy consumed relies on fossil fuels. The current efficiency of converting fossil fuel to usable forms of energy is in the 30% to 50% range for thermal power plants, and lowers for internal combustion engines, with the majority of energy lost in terms of waste heat. In theory, the waste heat can be recycled and generate electrical energy via Thermoelectric Effect.

Thermoelectric Effect converts heat directly into electricity. One of the main components of thermoelectric effect, the Seebeck effect, is the discovery that when two ends of a semiconductor are exposed to a temperature difference, there's a voltage difference generated across the semiconductor material. The magnitude of electric potential created by a temperature differential via Seebeck effect is measured by Seebeck coefficient (S), also known as thermopower. The Seebeck coefficient has units of V/K , and in practice a material with several $100\mu V/K$ is considered a good thermoelectric

material. Microscopically, Seebeck coefficient depends largely on the Density of States of a semiconductor material.

Photonic crystal is another application of band gap engineering, which essentially changes DOS and BS to create desired optical properties of a crystal. When a beam of photons incident on a crystal, if the incident photon's energy is roughly the same as the band gap, it gets absorbed by exciting an electron. On the other hand, if the photon's energy is nowhere near the energy of the gap (from the highest energy of the lower band to the lowest energy of the higher band), the photon gets scattered off. Therefore, all absorption of light happens near the band gap energy range. By changing the BS of a crystal (create gaps at degenerate energies, adjust height of the gap, etc) we can effectively change the absorption energy of the crystal and therefore its optical properties. In theory, the band gap acts as a "gate keeper" to allow either absorption or scattering of incident photons. Both photonic crystals and thermoelectric nanostructure can be modeled by the Kronig-Penney potential and the BS and DOS can be calculated in order to investigate in the properties of the materials.

However, analytical calculations of BS and DOS are only possible for very simple cases such as free electron. For purposes such as evaluating the Kronig-Penney model in one, two and three dimensions, the calculation must be done numerically. The objective of this project is to develop a program in Mathematica to calculate numerically the Density of States (DOS) and Band Structure (BS) of a crystal with given lattice parameters and potential.

2. CODING IN MATHEMATICA

2.1 Defining the lattice

The procedure of this project mainly contains a program that calculates the DOS for an arbitrary crystal with an arbitrary but known potential. The potential of the quantum dots are given, and the simulation is done assuming there's no electron-electron interaction. Appendix 1 is a complete code of the program written in Mathematica version 7.

The program starts by constructing lattice sites represented by points with real translation vectors a_1 , a_2 and a_3 . Using the following equations

$$\vec{b}_1 = \frac{\vec{a}_2 \times \vec{a}_3}{\vec{a}_1 \cdot \vec{a}_2 \times \vec{a}_3}, \quad \vec{b}_2 = \frac{\vec{a}_3 \times \vec{a}_1}{\vec{a}_2 \cdot \vec{a}_3 \times \vec{a}_1}, \quad \vec{b}_3 = \frac{\vec{a}_1 \times \vec{a}_2}{\vec{a}_3 \cdot \vec{a}_1 \times \vec{a}_2} \quad (2.1)$$

the basis of reciprocal lattice vectors b_1 , b_2 and b_3 can be calculated and a list of vectors

$$G_n = 2\pi(n_1 b_1 + n_2 b_2 + n_3 b_3) \quad (2.2)$$

can be generated using integers n_1 , n_2 and n_3 . G_n is linear combinations of reciprocal lattice basis vectors. In this case, a list of G_n vectors can be generated to contain all G-vectors from one point to another in the lattice. To make the calculation possible, a finite set of G vectors must be defined. In the case of this project, a Gcutoff value is defined to be the maximum amplitude of all G vectors. If a G vector has magnitude longer than Gcutoff, then it is not considered in the calculation.

In order to realize the elimination process described above, first a maximum value of n (n_{max}) must be defined by dividing Gcutoff by the smallest reciprocal lattice vector, then a list of triplets of n values are generated from $-n_{max}$ to n_{max} for each of n_1 , n_2

and n_3 . This list of integer triplets have the form of (n_1, n_2, n_3) , and the dot product of (n_1, n_2, n_3) and $2\pi*(b_1, b_2, b_3)$ constructs a list (named AllGnlist) of all G vectors to be considered. The subroutine “GnMagnitude” calls for (n_1, n_2, n_3) and returns the magnitude of the corresponding G vector. The command “Select” in Mathematica 7 is used in combination with the “GnMagnitude” subroutine to “scan” through all elements in AllGnlist and discard those with magnitudes larger than Gcutoff. The list of surviving elements is named Gnlist.

2.2 The Hamiltonian Matrix

Once we obtain a finite list of G-vectors, a Hamiltonian matrix can be formed in order to calculate the dispersion relation which determines the energy of an electron in the periodic potential. The Hamiltonian of an electron has two components: potential energy and kinetic energy, so the Hamiltonian matrix is the sum of two matrices: the potential energy matrix and the kinetic energy matrix.

The potential energy matrix is simply found by assuming the periodic potential of the lattice in real space is given. A Fourier Series of the potential in real space will yield the potential in momentum space, V_G , as a function of G-vectors. The terms in the potential energy matrix is then obtained by $V_g(G_i - G_j)$ where i is the row index and j is the column index.

For the kinetic energy matrix, we must briefly introduce Bloch’s theorem which exploits the periodic nature of the potential.

2.2.1 Bloch’s Theorem

Define a periodic potential,

$$V(r) = V(r + R) \quad (2.3)$$

Due to the translational invariance of the potential, Bloch theorem states that

$$\varphi_k(r + R) = e^{ik \cdot R} \varphi_k(r) \quad (2.4)$$

where states are labeled by corresponding wave vectors k . For the purpose of this project, the number of wave vectors determines the number of data points taken into consideration by the numerical calculation.

With the help of Bloch's theorem, we can write the kinetic energy matrix for each k as

$$\frac{(\vec{G}_i + \vec{k})^2}{2m} \quad (2.5)$$

Only the diagonal terms of the kinetic energy matrix are non-zero. Therefore, the Hamiltonian matrix is the potential energy matrix with its diagonal terms added by the corresponding diagonal terms from the kinetic matrix. In the program, H_0 represents the potential energy matrix, and becomes the Hamiltonian matrix later after adding the kinetic energy terms onto all of its diagonal elements. In order to construct the Hamiltonian matrix, a list of k -vectors is needed.

2.3 Calculate BS

Band Structure (BS) can be obtained by diagonalization of the Hamiltonian matrix for each k -vector, since the eigenvalues (energies) change as a function of k , and are different along different directions in k space. Traditionally, BS is plotted along axes of symmetry in a closed loop. For example, the BS for a square lattice in 2-D is plotted along the path of $a - b - c - a$ (Fig. 2.2.1a)

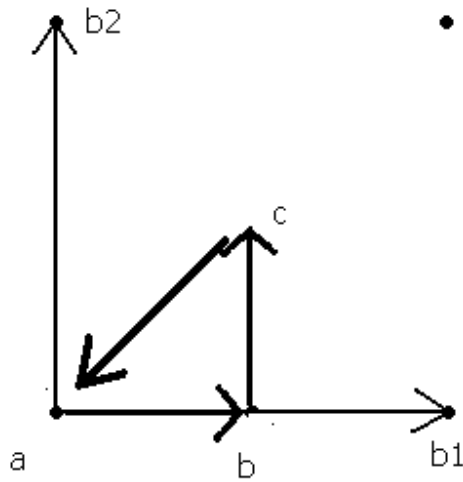


Fig. 2.3a, The direction of k-vectors moving along all axes of symmetry, with

$$\text{vectors } \vec{ab} = \frac{\vec{b}_2}{2}, \vec{bc} = \frac{\vec{b}_1}{2} \text{ and } \vec{ca} = \frac{(\vec{b}_2 + \vec{b}_1)}{2}$$

The subroutine “GeneratekMeshBS” creates a list with a specified population of k-vectors along a path defined by a starting point and an ending point. Each path yields a list of k-vectors, and all lists are joined together to form a single linear list of k-vectors. This list of k-vectors are then used to construct the Hamiltonian matrix which when diagonalized, yields the eigenvalues which are energies at the particular k point. The lowest energy band can be found by grouping together the lowest energies, and second lowest band is formed by the second lowest group of energies. A plot of all energy points shows clearly the band structure of the crystal defined in the initialization step. The routine is written in 3-D space but can be easily reduced to 2-D and 1-D by changing the length of lattice constants and confining the path of k-vectors in two or one dimension.

2.4 Calculate the Density of States (DOS)

DOS as a function of E can be calculated by doing a sum of all bands over all energies of k-vectors, using the following equation

$$DOS(E) = \frac{1}{N_k} \sum_k \delta(E - E_k) \quad (2.6)$$

where N_k is the total number of k-vectors covering the FBZ. Simply using the delta function gives a histogram-like count for the DOS, instead of yielding a function of E. Since all electrical/thermal properties of solids depends on DOS as a function of E, an simple histogram is not as useful. By replacing the delta function, $\delta(x)$, with the following Gaussian distribution

$$\delta(x) = \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} \quad (2.7)$$

where

$$x = E - E_k^\lambda$$

a dependence of energy is introduced. E_k^λ is the λ th band energy of state labeled by k. To reduce calculation time, and due to the fact that data points in the tail of the Gaussian distribution is much less important than the ones in the middle, any point that falls outside of 6σ (can be defined to any value as desired by user) of the Gaussian function are neglected from the calculation.

A different set of k-vectors are needed for calculation of DOS. This new list of k-vectors must cover all k-points within the First Brillouin Zone (FBZ), instead of merely running along the reciprocal lattice vectors. The subroutine GeneratekMeshDOS creates a list of k-vectors that fills the FBZ. Due to the translational invariance and symmetry of the system, DOS found within the FBZ is the same as the DOS found within a reciprocal lattice primitive cell. Therefore, to simply calculation, a mesh of k-vectors is generated

within a reciprocal lattice primitive cell. GeneratekMeshDOS subroutine divides reciprocal lattice vectors into user-specified number of segments, then creates integer multiples of each segment in all three directions and thus generate a “cube” array of vectors, but arranged in a linear list. This list of k-vectors is then used in construction of Hamiltonian matrix for calculating DOS.

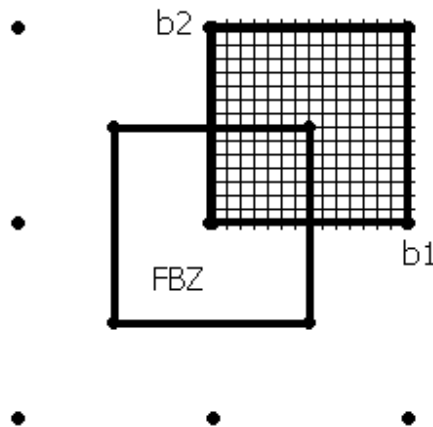


Fig.2.3b, k-mesh is generated within the primitive cell, with same area and thus same number of k-points as FBZ.

Energies at each k-point can be found by diagonalizing the Hamiltonian matrix. Taking the first several bands (again, user-specified) and summed over all k-vectors using the method described above, a DOS can be assembled as function of E. Plotting DOS using Mathematica’s built in plot command can be time consuming, due to large number of calculations involved. By using the “PlotDOS” subroutine, which takes number of points considered and width of the Gaussian function as inputs, the DOS is only calculated for a number equally spaced energy points and thus the calculation time is greatly reduced.

3. HOW TO USE THE PROGRAM

1. Open the file “Main.nb” with Mathematica
2. Evaluate the thread defining all subroutines
3. In a new Mathematica notebook file, type “Initialize” command, input

argument with crystal parameters in the following format:

$$\{\{a_{1x}, a_{1y}, a_{1z}\}, \{a_{2x}, a_{2y}, a_{2z}\}, \{a_{3x}, a_{3y}, a_{3z}\},$$

Number of k – vectors along each path for BS,

Number of k – vectors for DOS in b1 direction,

Number of k – vectors for DOS in b2 direction,

Number of k – vectors for DOS in b3 direction,

Cut off for G }

This command initiates the k-mesh arrays and the Fourier components of the potential.

4. The command “PlotBS[n]” plots the first n bands of energy. The vertical axis is energy, and the horizontal axis is merely a number count of k-vector, with no units (for example, 5 means the 5th k-vector in the list of “klistBS”).
5. The command “PlotDOS[Emin, Emax, *energy step*, *sigma*]” plots the DOS using the specified interval with energy between Emin and Emax, energy step defines the interval between sampling points along the x-axis, and the width of Gaussian curve being sigma. The vertical axis is number of states and the horizontal axis is energy. By default the first 10 bands are considered in the calculation. If there are less than 10 bands, then all the bands are considered.
6. Units: This program is written with atomic units in mind ($\hbar = m = e = 1$)

The program is written in version 7.0 and therefore may not be compatible with older versions due to slightly different syntax and formatting rules.

4. TEST WITH KNOWN EXAMPLES

In order to test the functionality of the program, I ran the following tests:

4.1 Testing BS

Taking a square lattice: $a_1 = (2\pi, 0, 0)$, $a_2 = (0, 2\pi, 0)$, $a_3 = (0, 0, 2\pi/100)$

which is very small compared to a_1 and a_2 , to ensure b_1 is (1,0,0), and b_2 is (0,1,0) and $b_3=100$ is very large and therefore even the smallest multiple of b_3 is still omitted by G-cutoff defined to be 2. There are 13 G-vectors, which constructs the following 13 by 13 matrix:

1.99	0	0	0	0	0	0	0	0	0	0	0	0
0	0.99	0	0	0	0	0	0	0	0	0	0	0
0	0	0.5	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1.99	0	0	0	0	0	0	0	0
0	0	0	0	0	0.5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.51	0	0	0	0	0
0	0	0	0	0	0	0	0	2.01	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0.51	0	0
0	0	0	0	0	0	0	0	0	0	0	1.01	0
0	0	0	0	0	0	0	0	0	0	0	0	2.01

Table 4.1 The Hamiltonian Matrix for free electron at $k=(0,0,0)$

and the calculated BS is as follows

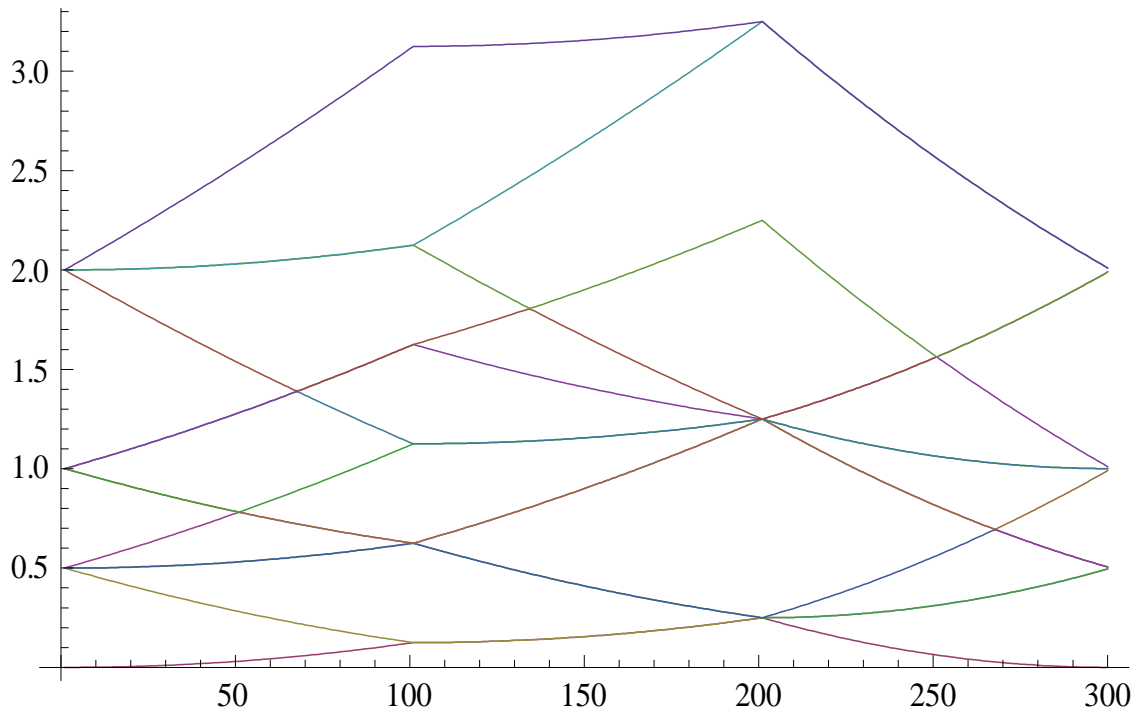


Fig 4.1a. The BS of a two dimensional crystal lattice under zero potential (2-D free electron) along the path shown in Fig. 2.3a.

If we put a potential in the lattice, thus add non-zero off diagonal elements to the matrix, then we should see the splitting of bands. The potential $V = 0.2e^{G^2}$ adds non-zero off diagonal terms to the matrix, and should cause the degenerate bands to split. The following figure confirms the result.

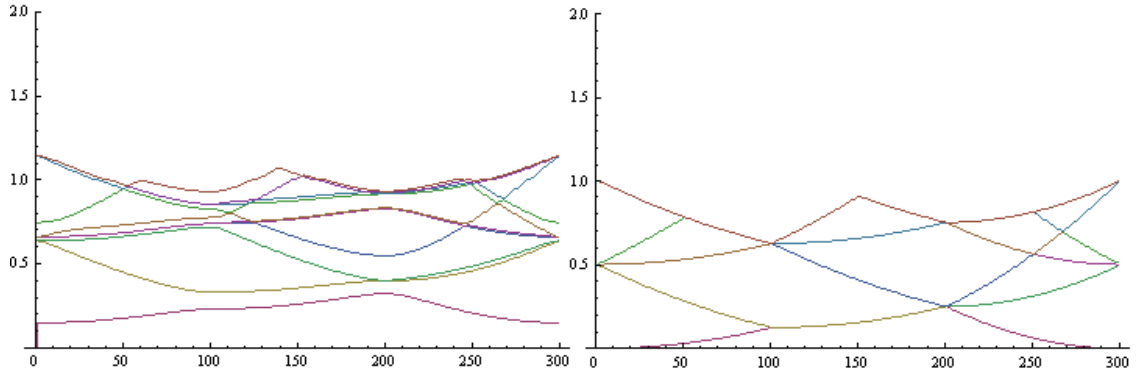


Fig 4.1b. Left: Band splitting caused by non-zero diagonal terms under influence of a potential. The band splitting is visible when compared with the plot with zero-potential.

Right: Same plot with zero-potential as comparison.

4.1 Testing DOS

The DOS of a free electron in one, two and three dimensions are tested using the program and compared with the analytical result.

4.1.1 Testing in 1-D

Taking a square lattice: $a_1 = (2\pi, 0, 0)$, $a_2 = (0, 2\pi/100, 0)$, $a_3 = (0, 0, 2\pi/100)$ which is very small compared to a_1 and a_2 , to ensure b_1 is $(1,0,0)$, and b_2 is $(0,1,0)$ and b_3 is very large and therefore even the smallest multiple of b_3 is still omitted by G-cutoff defined to be 4.5. Under no potential, with 25 k-vectors along b_1 direction and 1 k-vector along b_2 and b_3 direction, the one dimensional DOS of a free electron is plotted in the following figure:

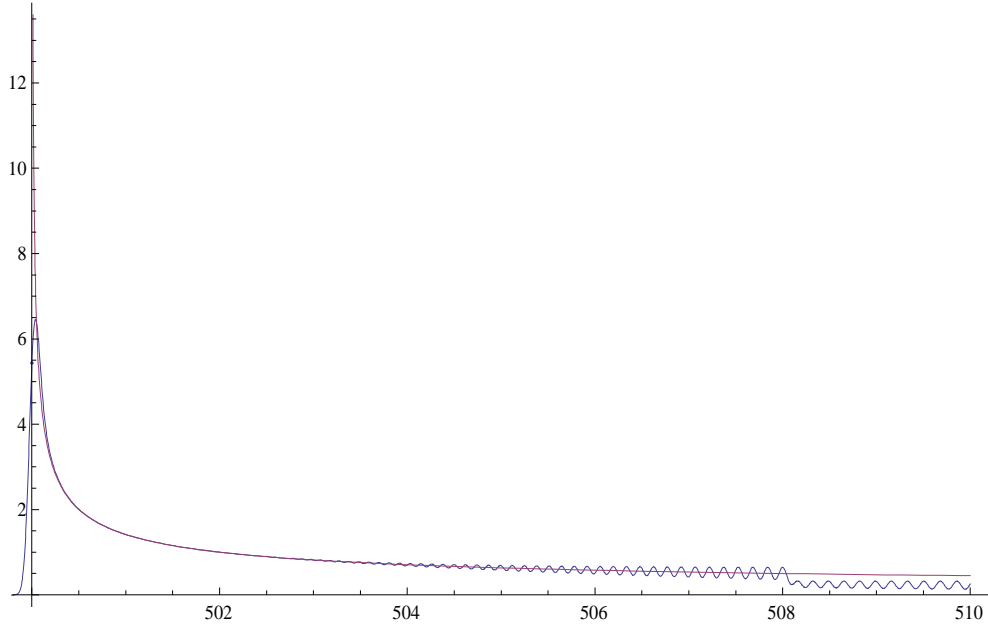


Fig. 4.1.1 Calculated DOS of 1-D free electron super imposed with $\sqrt{2}E^{-\frac{1}{2}}$ function. The wave towards right is due to the Gaussian used in the calculation. The vertical axis is the DOS and the horizontal energy is energy. The width of the Gussian function is 0.05.

The oscillating curve is the result of the calculation, and the non-oscillating curve is the analytical $DOS = \sqrt{2} \frac{1}{\sqrt{E-500}}$ plotted as a comparison. The analytical result is obtained by the following means:

Since the density of state for free electron is $2 \left(\frac{L}{2\pi}\right)^n$ multiplied by the DOS calculated by this program, where n is the number of dimensions, and the factor of 2 accounts for the spin states of an electron. The analytical DOS is given in Kittel's Introduction to Solid Physics book (Kittel, page 141):

$$DOS(E) = \frac{dN}{dE} \quad (4.1)$$

where N is number of available states and E is the energy. Equation 4.1 can be written as

$$DOS(E) = \frac{dN}{dk} \frac{dk}{dE} \quad (4.2)$$

Since $N = 2 \frac{L}{2\pi} k$ in 1-D, equating the calculated DOS (let's call it MyDOS for now)

and replace n with 1 (for 1-D) the equation looks like

$$2 \left(\frac{L}{2\pi} \right)^1 \text{MyDOS} = 2 \frac{L}{2\pi} \frac{\sqrt{2}}{\sqrt{E}} \quad (4.2)$$

Under atomic units, m and \hbar are 1 and L in the equation is also 1 since the length of reciprocal lattice constants are all 1. Simplify equation 4.2, we obtain

$$\text{MyDOS} = \sqrt{2} E^{-\frac{1}{2}} \quad (4.2)$$

if done analytically. This function is same graph with the calculated DOS, with a shift of 500 to the right since the lowest eigenvalue is 500. These two curves overlap for the most part indicates that the calculated DOS is indeed a valid approximation of the exact DOS of a 1-D free electron. The inaccuracy of DOS increases as energy goes up, because the calculation only considered the first ten bands in order to save calculation time. In this particular plot, the DOS became very wavy towards high energies and the oscillation amplitude decreased by half due to insufficient data at high energy levels.

4.1.2 Testing in 2-D

In 2-D, the analytical result of DOS should be a constant for free electrons. The following result is a plot of calculated value with the same parameter as the 1-D case, except now $a_2 = (0, 2\pi, 0)$, there are only 7 k-vectors along each b-vector, and plotted from 1 to 2 in energy:

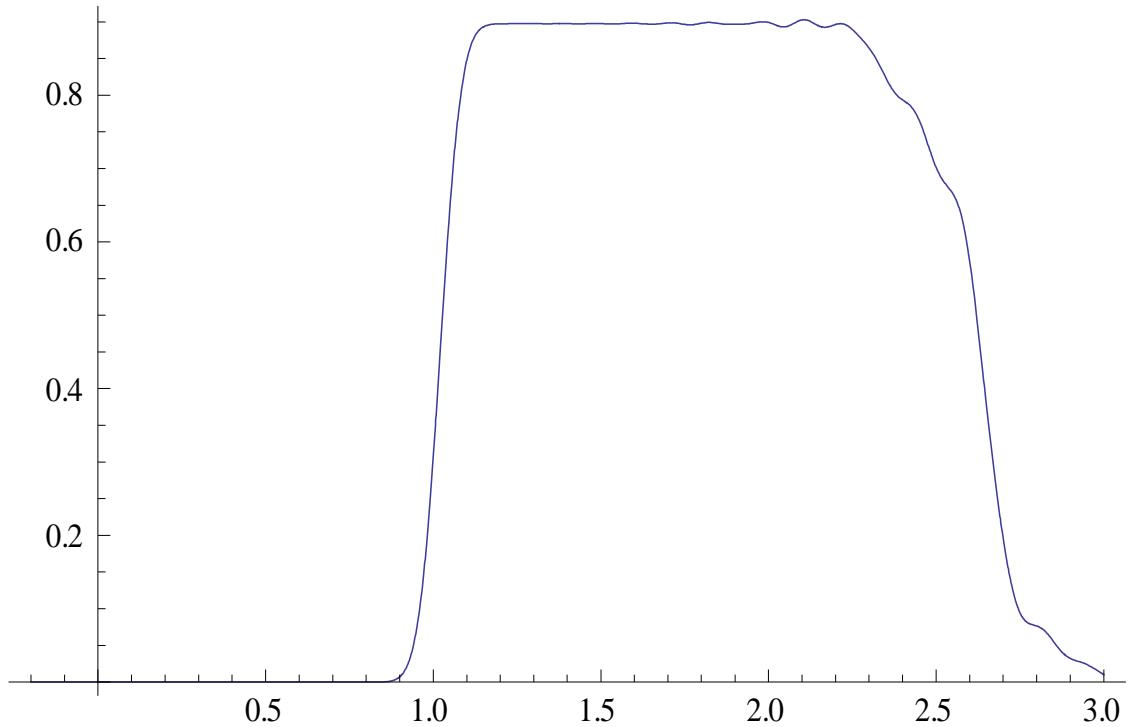


Fig 4.1.2 The plot of calculated DOS for 2-D free electron. The horizontal axis is energy (in atomic units) and the vertical axis is the DOS. There's no states below 1 due to the fact that there are too few k-vectors used in the calculation. The DOS becomes inaccurate past $E=2$ because there aren't enough eigenvalues beyond that point. Note that the top of the plot is flat which serves as a good indicator. The width of the Gaussian is 0.05.

In theory, the DOS should be constant for all energies. The higher energies were omitted in this calculation in order to save computation time, and hence the tail on the right. The tail on the left is due to the fact that there are too few k-points in the mesh in order to reduce computation time, but resulted no eigenvalues smaller than 1. The fact that the plot is constant within its accurate range (not affected by the decay, that is) indicates that the program is calculating correctly the DOS of 2-D free electron states as a function of E .

4.1.2 Testing in 3-D

In 3-D, the analytical result of DOS should look like a function of $E^{\frac{1}{2}}$. The following result is a plot of calculated value with the same parameter as the 1-D and 2-D case, except now $a_3 = (0, 2\pi, 0)$ and plotted from 0 to 1 in energy:

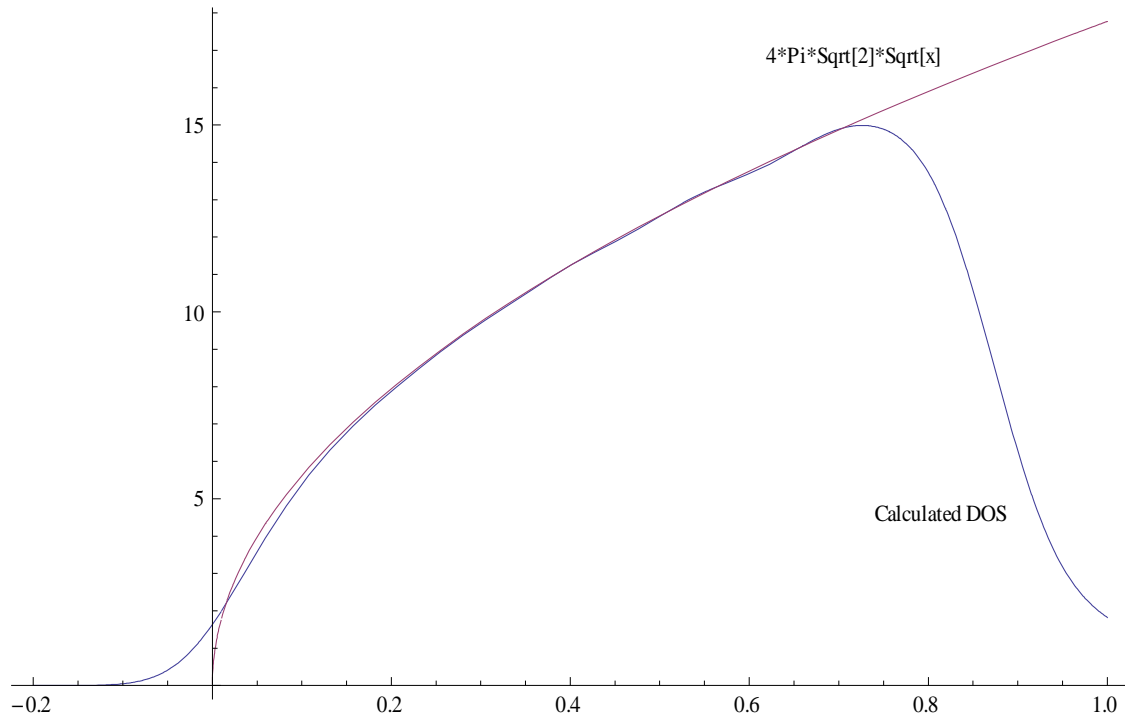


Fig 4.1.2 The plot of calculated DOS for 3-D free electron. The horizontal axis is energy (in atomic units) and the vertical axis is the DOS. The analytical DOS is also plotted and labeled.

In theory, the DOS should be proportional to \sqrt{E} . The higher energies were omitted in this calculation in order to save computation time. The analytical result is obtained by the following derivation:

Since the density of state for free electron is $2\left(\frac{L}{2\pi}\right)^n$ multiplied by the DOS calculated by this program, where n is the number of dimensions, and the factor of 2

accounts for the spin states of an electron. The result given by Kittel's Introduction to Solid Physics book (Kittel, page 140) is in the following form

$$DOS(E) = \frac{V}{2\pi^2} \left(\frac{2m}{\hbar} \right)^2 E^{\frac{1}{2}} \quad (4.1)$$

therefore, equating the calculated DOS (let's call it MyDOS for now) and replace n with 3 (for 3-D) the equation looks like

$$2 \left(\frac{L}{2\pi} \right)^3 \text{MyDOS} = \frac{V}{2\pi^2} \left(\frac{2m}{\hbar} \right)^2 E^{\frac{1}{2}} \quad (4.2)$$

Under atomic units, m and \hbar are 1 and L in the equation is also 1 since DOS is only calculated for the unit volume (and therefore V=1 as well). Simplifying equation 4.2, we obtain

$$\text{MyDOS} = \pi 4 \sqrt{2} E^{\frac{1}{2}} \quad (4.2)$$

if done analytically, which is plotted in the same graph with the calculated DOS. These two curves overlap for the most part indicating that the calculated DOS is indeed a valid approximation of the exact DOS of a 3-D free electron gas. The tail on the left side of the plot is caused by the broadening Gaussian.

5. CONCLUSION

The tests conducted in part 4 confirm the result of this program is a good approximation of the exact DOS and BS of the system specified. Knowing that the program works, one can find the crystal potential by Fourier transforming the step potential in the Kronig-Penney model with different lattice parameters, and plot BS and DOS for each lattice. By specifying confinements in one, two and three dimensions, one can essentially find the arrangement of lattice sites that is desired for a specific electrical or optical property (which depends largely on DOS and BS). Physically, this models

change of the arrangements of nanostructure such as quantum dots (3-D confinement), quantum wires (2-D confinement) and quantum wells (1-D confinement).

6. ACKNOWLEDGEMENTS

I would like to thank Keivan Esfarjani for his guidance and patience during the past several months of being my Technical Advisor of this project. His effort and time spent in teaching me everything I need to know in order to complete this project will always be appreciated.

I'd also like to thank Peter Young, Zack Schlesinger and Paul Graham for their help and advice, and Dave Belanger providing tips on writing and formatting of the paper.

Last but not least, I'd like to thank the Undergraduate Advisor Tery Penninnton, my colleagues Carlos Hernandez, Saahil Shenoy, Nhi Hua and all other great people around me. Without their help and words of encouragement, I could not have done this project.

7. REFERENCES

1. Ziman, J.M. (1964), Principles of The Theory of Solids, Cambridge: The Syndics of The Cambridge University Press
2. <http://physics.ucsc.edu/~keivan/teaching.html>
3. P. J. Mohr, B. N. Taylor, and D. B. Newell, Rev. Mod. Phys **80**(2), 633-730(2008)
4. Kittel, C. (2005), Introduction to Solid State Physics - 8th edition, John Wiley & Sons, Inc

Appendix 1

Complete Source Code in Mathematica

```
(*Declare Lattice Constants*)
(*Lattice constants a1, a2, and a3*)
GenerateLattice[{x10_,y10_,z10_},{x20_,y20_,z20_},{x30_,y30_,z30_}]:=
Module[{x1=x10, x2=x20, x3=x30, y1=y10, y2=y20, y3=y30, z1=z10, z2=z20, z3=z30},
a1={x1,y1,z1};
a2={x2,y2,z2};
a3={x3,y3,z3};
omega=Dot[a1,Cross[a2,a3]];
b1=2*Pi*Cross[a2,a3]/omega;
b2=2*Pi*Cross[a3,a1]/omega;
b3=2*Pi*Cross[a1,a2]/omega;
Return[{b1,b2,b3}];
```

```
(*Generate k-mesh BS*)
GeneratekMeshBS[mesh0_,{startx0_,starty0_,startz0_},{endx0_,endy0_,endz0_}]:=
Module[{mesh=mesh0,startx=startx0,starty=starty0,startz=startz0,endx=endx0,endy=end
y0,endz=endz0},
Clear[i];
klist0={};
For[i=0,i<=mesh-1,i++,
klist0=Append[klist0,{startx,starty,startz}+i/mesh*({endx,endy,endz}-
{startx,starty,startz})]
];
Return[klist0]
]
```

```
(*Generate k-mesh DOS*)
GeneratekMeshDOS[meshDOS10_,meshDOS20_,meshDOS30_]:=
Module[{meshDOS1=meshDOS10,
meshDOS2=meshDOS20,meshDOS3=meshDOS30},
Clear[i];
Clear[j];
Clear[k];
klistDOS={};
```

```

For[i=0,i<=meshDOS1-1,i++,
For[j=0,j<=meshDOS2-1,j++,
For[k=0,k<=meshDOS3-1,k++,
klistDOS=Append[klistDOS,(i+0.2)*b1/meshDOS1+(j+0.3)*b2/meshDOS2+(k+0.1)*b3/
meshDOS3];
]
]
]
]

```

```

(*Generate Gn*)
GenerateAllGn[Gcutoff_]:=
Module[{nmax=Ceiling[Gcutoff/Min[{Sqrt[b1[[1]]^2+b1[[2]]^2+b1[[3]]^2],
Sqrt[b2[[1]]^2+b2[[2]]^2+b2[[3]]^2],Sqrt[b3[[1]]^2+b3[[2]]^2+b3[[3]]^2}]}],
AllGnlist={};
For[n1=-nmax,n1<=nmax,n1++,For[n2=-nmax,n2<=nmax,n2++,For[n3=-
nmax,n3<=nmax,n3++,AllGnlist=Append[AllGnlist,{n1,n2,n3}]]];
Return[AllGnlist]

```

```

(*Calculate Magnitude of Gn*)
GnMagnitude[{{n1G0_,n2G0_,n3G0_}}]:=
Module[{n1G=n1G0,n2G=n2G0,n3G=n3G0},
rvector=n1G*b1+n2G*b2+n3G*b3;
doMagnitude[{Gx_,Gy_,Gz_}]:=Sqrt[Gx^2+Gy^2+Gz^2];
doMagnitude[rvector](*;
Return[doMagnitude[rvector]];*)
]

```

```

(*Make a list of all Gn less than Gcutoff*)
GenerateGn[Gcutoff0_]:=
Module[{Gcutoff=Gcutoff0},
Gnlist=Select[AllGnlist,GnMagnitude[{}]<=Gcutoff&];
NumOfElements=Length[Gnlist];(*NumOfElements displays the total number of
elements in the list Gnlist*)
Return[Gnlist]
"\"There are \"NumOfElements \" elements in the list.\"
]

```

```

(*Geneate H0 Matrix*)
GenerateH0Matrix[Gnlist0_]:=
Module[{Gnlist=Gnlist0},
For[i=1,i<= NumOfElements,i++,
For[j=1,j<= NumOfElements,j++,
H0[i,j]=VofG[Gnlist[[i]]-Gnlist[[j]]];
];

```

```

];
]

(*Diagonalize H Matrix for DOS*)
EpsilonDOS[lambda0_,k0_]:=
Module[{lambda=lambda0,k=k0},
GenerateH0Matrix[Gnlist];
For[i=1,i<=Length[Gnlist],i++,
H0[i,i]=H0[i,i]+Shift+Dot[(Gnlist[[i]]+klistDOS[[k]]),(Gnlist[[i]]+klistDOS[[k]])/2
];
MatrixH={};
For[i=1,i<=Length[Gnlist],i++,
H0i={};
For[j=1,j<=Length[Gnlist],j++,
H0i=Append[H0i,H0[i,j]];
];
MatrixH=N[Append[MatrixH,H0i]]
];
(*Print[MatrixH0//MatrixForm];*)
(*For the 1st 20 bands*)
EpsilonValuesDOS=Sort[Eigenvalues[MatrixH, -NumberofBands]-Shift];
Return[EpsilonValuesDOS[[lambda]]]
]

```

```

(*Assemble all Eigenvalues*)

```

```

(*Calculate DOS*)
Clear[DOS];
Delta[x_,sigma_]:= (1/Sqrt[2*Pi*sigma^2])*Exp[-x^2/(2*sigma^2)]
DOS[e00,sigma0_]:=
Module[{e0=e00, sigma=sigma0},
sum=0.;
Clear[k];
For[k=1,k<=Length[TotalEigenValuesDOS],k++,
If[Abs[e0-TotalEigenValuesDOS[[k]]]<6*sigma,
sum=sum+Delta[(e0-TotalEigenValuesDOS[[k]]),sigma];
];
];
Return[DOS0=(1/Length[klistDOS])*sum];
];

```

```

(*Plot DOS*)
PlotDOS[minEnergy0_,maxEnergy0_,interval0_, $sigma0_]:=
Module[{minEnergy=minEnergy0,maxEnergy=maxEnergy0,interval=interval0,
$sigma=$sigma0},
Clear[k];

```

```

TotalEigenValuesDOS={ };
For[k=1,k<= Length[klistDOS],k++,
EpsilonDOS[1,k];
If[Element[k/10,Integers],Print[k]];
TotalEigenValuesDOS=Join[TotalEigenValuesDOS,EpsilonValuesDOS]
];
Print[Plot[DOS[x,$sigma],{x,minEnergy,interval,maxEnergy}]]
];

(*Diagonalize H-Matrix for BS*)
EpsilonBS[lambda0_,k0_, nofband0_]:=
Module[{lambda=lambda0,k=k0,nofband=nofband0},
GenerateH0Matrix[Gnlist];
For[i=1,i<= Length[Gnlist] ,i++,
H0[i,i]=H0[i,i]+Shift+Dot[(Gnlist[[i]]+klistBS[[k]]),(Gnlist[[i]]+klistBS[[k]])/2
];
MatrixH={ };
For[i=1,i<=Length[Gnlist],i++,
H0i={ };
For[j=1,j<=Length[Gnlist],j++,
H0i=Append[H0i,H0[i,j]];
];
MatrixH=N[Append[MatrixH,H0i]]
];
(*Print[MatrixH0//MatrixForm];*)
(*For the 1st 20 bands*)
EpsilonValuesBS=Sort[Eigenvalues[MatrixH,-NumberofBands]];
Return[EpsilonValuesBS[[lambda]]-Shift]
]

(*Plot BS*)
PlotBS[NumOfBands0_]:=
Module[{NumOfBands=NumOfBands0},
EpsilonBS[1,1,NumOfBands];
ListLinePlot[Table[{k,EpsilonBS[m,k,NumOfBands]},{m,0,NumOfBands},{k,0,Length[
klistBS]}],PlotRange->Automatic(*{50000/2,50013/2}*)]
];

(*Initialization*)
Initialize[{$x10_,$y10_,$z10_},{x20_,$y20_,$z20_},{x30_,$y30_,$z30_},$meshBS0
,$meshDOS10_,$meshDOS20_,$meshDOS30_,$Gcutoff0_]:=Module[{$x1=$x10,$
y1=$y10,$z1=$z10,$x2=$x20,$y2=$y20,$z2=$z20,$x3=$x30,$y3=$y30,$z3=$z30,$
meshBS=$meshBS0,$meshDOS1=$meshDOS10,$meshDOS2=$meshDOS20,$mesh
DOS3=$meshDOS30,$Gcutoff=$Gcutoff0},
GenerateLattice[{$x1,$y1,$z1},{x2,$y2,$z2},{x3,$y3,$z3}];

```



```

klistBS={};
klistBS1=GeneratekMeshBS[$meshBS,{0,0,0},b1/2];
klistBS2=GeneratekMeshBS[$meshBS,b1/2 ,(b1+b2)/2];
klistBS3=GeneratekMeshBS[$meshBS,(b1+b2)/2 ,{0,0,0}];
klistBS=Join[klistBS1,klistBS2,klistBS3];

GeneratekMeshDOS[$meshDOS1,$meshDOS2,$meshDOS3];
GenerateAllGn[$Gcutoff];
GenerateGn[$Gcutoff];

Clear[VofG];
d=2*Pi/5*Sqrt[Dot[a1,a1]];
l=d/5;
(*VofG[{n1_,n2_,n3_}]:=0.2 Exp[-(n1 n1+n2 n2 +n3 n3)];*)
(*VofG[{n1_,n2_,n3_}]:=If[n2==0&& n3==0,
Return[Sin[(n1+0.00001)*d]/(n1+0.00001)*Exp[-n1*n1*1/2]],,
Return[0];
];*)
(*VofG[{n1_,n2_,n3_}]:=Cos[n1];*)
VofG[{n1_,n2_,n3_}]:=0;

Clear[Shift];
Shift=5(Dot[b1,b1]+Dot[b2,b2]+Dot[b3,b3])/2;
NumberOfBands=Min[Length[Gnlist],10];

EpsilonDOS[1,1];
];

```