Solid State I - Final Project

Robert Rosati

May 12 2015

Abstract

This project sought to explore electronic band structure theory and practical methods of band structure computation. From first principles, the existence of electronic band structure is shown and, after a series of approximations, an algorithm developped to find it numerically. I further refined the algorithm and coded it in Python, and used the code to find the band structure of Aluminum in agreement with the text and reference data.

1 Schrödinger's Equation in a Lattice

Band structure theory arises from considering Schrödinger's equation for an electron in a Bravais lattice. In Fourier space, Schrödinger's equation can be expressed:

$$\left(\frac{\hbar^2 q^2}{2m_e} - \varepsilon\right)\psi(\vec{q}) + \sum_{\vec{K}} \psi(\vec{q}-\vec{K})U_{\vec{K}} = 0 \tag{1}$$

where the $\vec{K}$ are reciprocal lattice sites, $U_{\vec{K}}$ the potential in Fourier space, and the $\vec{q}$ electron wavenumbers. Or in Dirac notation,

$$\begin{gathered} H = \sum_{\vec{q}'} |\vec{q}'\rangle \frac{\hbar^2 q^2}{2m_e} \langle \vec{q}'| + \sum_{\vec{q}'\vec{K}'} |\vec{q}'\rangle U_{\vec{K}'} \langle \vec{q}' - \vec{K}'| \\ \langle \vec{q}| H |\psi\rangle = \varepsilon |\psi\rangle \end{gathered} \tag{2}$$

Note translating $\vec{q}$ by a reciprocal lattice vector $\vec{q}+\vec{K}$ leaves the Fourier components of

ψ unchanged. To see this, following Marder, use equation (1):

$$(\varepsilon^0_{\vec{q}+\vec{K}} - \varepsilon)\psi(\vec{q}+\vec{K}) + \sum_{\vec{K}'} \psi(\vec{q}+\vec{K}-\vec{K}')U_{\vec{K}} = 0$$
$$(\varepsilon^0_{\vec{q}+\vec{K}} - \varepsilon)\psi(\vec{q}+\vec{K}) + \sum_{\vec{K}'} \psi(\vec{q}-\vec{K}')U_{\vec{K}'+\vec{K}} = 0$$

where $\varepsilon^0_q = \frac{\hbar^2 q^2}{2m}$ and the second line relabels the summation index $\vec{K}'$ to $\vec{K}'-\vec{K}$. The second line is a different equation than (1) with a different solution, but involves the same Fourier components of ψ. Therefore multiple solutions to Schrödinger's equation exist, with independent energy eigenvalues. These are termed bands, and their eigenvalues and eigenvectors are conventionally labelled ε_{nk} and ψ_{nk} respectively.

By the argument above, any two $\vec{q}$ that differ precisely by a reciprocal lattice vector are physically indistinguishable from two $\vec{q}$ at the same lattice site but in different bands (i.e. $\psi_{n,\vec{k}+\vec{K}} = \psi_{n',\vec{k}}$ with $n \neq n'$). It is customary to take all components of $\vec{k}$ in the interval $[0, 2\pi/a]$, between the origin and the first reciprocal lattice site in the all-positive-coordinates direction. This is termed the first Brillouin zone.

2 Band Structure

Electronic band structure is capable of predicting a wide range of material properties if known. The density of energy states is related to $\frac{d\varepsilon}{dk}$, and is directly related to specific heat among other properties. Perhaps the most easily visible information from a band structure plot is whether or not the material is a conductor or an insulator. Though not always preserved by the approximations used to make band structure computations tractable, the gap between a material's highest energy level below the Fermi energy and the lowest level above it in large part determines the conductivity of the material. A larger gap makes the material more insulating, while a smaller or nonexistent gap allows more electrons to move freely. Given an effective potential for the outermost electron, the Fermi energy can be computed as $\frac{1}{\Omega}U_{\text{eff}}(\vec{k}=0) = -\frac{2}{3}\varepsilon_F$, where Ω is the volume of a unit cell [4].

3 Computation

The potential in (1) has, in gerneral, $O(N^{ZN})$ electron-electron interactions, where Z is the atomic number of the lattice material and N the number of lattice sites in the simulation. A common approximation is to assume that only the valenence electron at each lattice site has any kinetic energy. This approximation is particularly good for materials like alkali metals, where the inner electrons are much more tightly bound the valence electron. The effect of the core electrons is to screen the charge of the nucleus, and the valence electrons can be modeled as under the influence of a so-called pseudopotential, which effectively dampens the depth of the Coulomb $-k/r$ potential near the origin. See figure 1. This approximation brings the potential complexity to $O(N^N)$; however computing all valence electron - valence electron interactions is still outside the realm of tractability. Treating each electron indpendently brings the complexity to $O(N^2)$, solidly within the reach of modern computers for realistic materials. Perhaps somewhat surprisingly, this approximation gives results in at least qualitative agreement for most materials. Materials with strong electron-electron coupling (like NiO and CuO) are an exception [4].

The efficiency of band structure calculation is highly dependent on the basis representation of H and ψ. While equation (1) describes a wavefunction in a plane wave basis, in general this does not guarantee a convergent solution with finitely many Fourier components. Other, more elaborate bases (such as linear-augmented plane waves, LAPW) can provide convergence with fewer components. Such bases are outside the scope of this work and will not be further discussed.

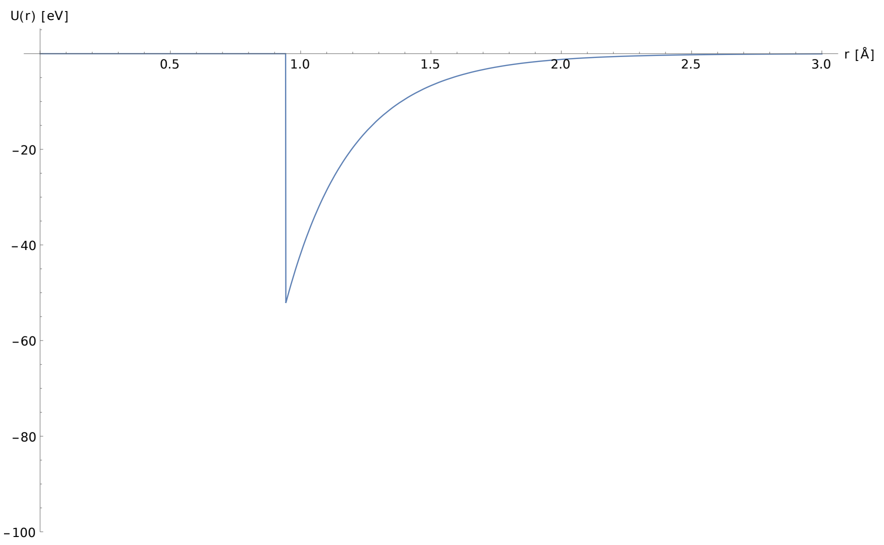

Figure 1: The Ashcroft or "empty-core" pseudopoential $U(r) = U_0 e^{-r/d} \frac{\Omega}{4\pi d^3} \frac{d}{r} \theta(r - R_c)$ with the parameters U_0, d, and R_c fit from aluminum experimental data. Plot generated with *Mathematica*.

3.1 Plane Waves

The following arguments assume the wavefunction has the same form as in equation (2) (plane-wave basis). If the Fourier coefficients of ψ drop to some negligibly small value at high frequencies, this problem becomes finite-dimensional.

Perhaps the most straightforward method of solving equation (1) is to attack the k-space Hamiltonian directly. Given the Hamiltonian, H, an eigenvector can be found by the following process (as described in Marder, p.273)[4]. Guess an eigenstate with N Fourier components, $\vec{\phi_1}$, and compute $H\vec{\phi_1} = \vec{\phi_1}'$. Replace $\vec{\phi_1}$ by $\vec{\phi_1}'$ and compute $\vec{\phi_1}''$. After r iterations of this, $\vec{\phi_1}^{(r)} = \sum_{i=1}^{N} \varepsilon_i^r \hat{\psi}_i (\hat{\psi}_i \cdot \vec{\phi_1})$, where the ε_i and $\hat{\psi}_i$ are the eigenvalues and eigenvectors. If H is offset so that the lowest eigenvalue has the largest amplitude, then the $i = 1$ term will dominate the sum as $r \to \infty$, assuming the randomly chosen $\vec{\phi_1}$ is not orthogonal to $\hat{\psi}_1$, i.e. this process has the effect of amplifying the component of $\vec{\phi_1}$ parallel to the lowest eigenvector. Thus $\vec{\phi_1}^{(r)}$ is proportional to only the first eigenvalue and eigenvector for sufficient r, and these can be stored for later use. By using the Gram-Schmidt process, one can obtain a $\vec{\phi_2} = \vec{\phi_1} - \hat{\psi}_1(\hat{\psi}_1 \cdot \vec{\phi_1})$ orthogonal to $\hat{\psi}_1$, and repeating the above steps will

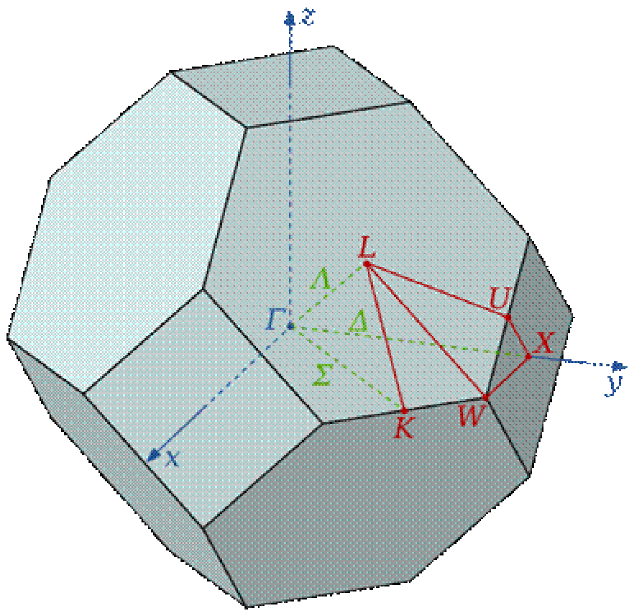

Figure 2: The first Brillouin zone of face-centered-cubic reciprocal lattice, with high symmetry points marked. In terms of $2\pi/a$, the points used in the band structure plot below are $L = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, $\Gamma = (0, 0, 0)$, $X = (0, 1, 0)$, and $U = (\frac{1}{4}, 1, \frac{1}{4})$ [3].

give the second eigenvalue and eigenvector of the Hamiltonian. This can be repeated until the entire set of N eigenvalues has been exhausted.

One aid in this calculation is that the convolution in the potential term may be replaced by a multiplication in Fourier space. That is,

$$\sum_{\vec{q},\vec{K}} U_{\vec{K}}\phi(\vec{q} - \vec{K}) = N^{-3}\mathcal{F}^{-1}\left\{\mathcal{F}\left\{U_{\vec{K}}\right\} \times \mathcal{F}\left\{\phi(\vec{q})\right\}\right\}$$

where $\mathcal{F}$ is the Fourier transform.

3.2 Linear Algebra Optimization

The above method is relatively inefficient – energy eigenvalues are only extracted after tediously computing the eigenfunctions through a slowly-converging series of repeated applications of H. If only the eigenvalues of the Hamiltonian are desired, then they can be found through a more direct method, such as writing the Hamiltonian as a matrix, $\mathcal{H}$, and solving its characteristic polynomial: $\det(\mathcal{H} - \lambda I_N) = 0$. In the plane wave basis this matrix can be written

$$\begin{aligned}\mathcal{H}(\vec{q}', \vec{q}) = \langle \vec{q}' | \, H \, | \vec{q} \rangle &= \delta_{\vec{q},\vec{q}'} \frac{\hbar^2 q^2}{2m_e} + U_{\vec{K}'} \delta_{\vec{K}',\vec{q}'-\vec{q}} \\ &= \delta_{\vec{q},\vec{q}'} \frac{\hbar^2 q^2}{2m_e} + U_{\vec{q}'-\vec{q}}\end{aligned} \tag{3}$$

where H is given by equation (2) and the $\vec{q}$s are some vector in the first Brillouin zone plus a reciprocal lattice vector $\vec{q}_n = \vec{k} + \vec{K}_n$. Note that this matrix is symmetric if and only if the potential is symmetric. The pseudopotential in figure 1 depends only on the magnitude of $\vec{K}$, so its corresponding $\mathcal{H}$ is symmetric (and entirely real, therefore also Hermitian). While $\mathcal{H}$ is still expressed in the plane wave basis, this approach avoids some of its inherent disadvantages by avoiding the eigenvector computation altogether. It does require more memory, as the Hamiltonian is stored as a matrix and not simply as a function. The memory penalty of storing this matrix should be no more than $8 \text{ bytes} \times N^2$ for a double-precision floating point array. For a run of $N = 12^3 = 1728$ lattice sites, the memory penalty is less than 24 MB – negligible on most modern hardware.

Referencing equation (3), evaluating the Hamiltonian in a different place in k-space only changes the kinetic terms, which lie along the diagonal. My code takes advantage of this and reuses off-diagonal terms when tracing the band structure plot path.

One advantage of this approach is that quite efficient, parallelized Hermitian matrix eigenvalue solvers exist in most scientific computing environments. For my implementation I used Python's `scipy.linalg.eigvalsh`, a wrapper around the Fortran LAPACK library's `geev` routine [2].

4 Code and Plots

Included below is a minimum subset of code required to produce figure 3, using the matrix eigenvalue technique.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import numpy as np
from scipy import linalg as LA
```

```python
import matplotlib.pyplot as plt
import matplotlib

# unit conversions
eV_to_Ryd = 1/13.6
c = 3.80998 # hbar**2 / (2*m) in  Angstroms^2 * eV

#pseudopotential parameters for Aluminum
d = 0.350 # angstroms
rc = 0.943 # angstroms
U0 = -31.30 # eV

# U(K) modified from Marder's (10.38) to give its limit at K=0
def U(K):
    Kn = LA.norm(K)
    if Kn > 0:
        return U0 * np.exp(-rc/d) * ((np.sin(rc*Kn) / (d*Kn * ((d*Kn)**2 + 1)) +
            (np.cos(Kn*rc))/((d*Kn)**2 + 1)))
    else:
        return U0*np.exp(-rc/d)*(rc/d+1)
vector_U = np.vectorize(U) # this function applies U to an array of K's

# Al is an fcc lattice with a spacing of a = 4.05 angstroms
a = 4.05 # angstrom
(a1,a2,a3) = 0.5 * a * np.array([[1,1,0],[1,0,1],[0,1,1]])

# the lattice's reciprocal vectors are given by Marder's equation 3.24
b1 = 2 * np.pi * np.cross(a2,a3) / a1.dot(np.cross(a2,a3))
b2 = 2 * np.pi * np.cross(a3,a1) / a2.dot(np.cross(a3,a1))
b3 = 2 * np.pi * np.cross(a1,a2) / a3.dot(np.cross(a1,a2))

# index shuffle to ensure we have negative K
def l(ji,N):
    return ji-N if ji+1 > (N+1)/2 else ji

def generate_lattice(N):
    K = np.empty([N,N,N], dtype=np.ndarray)
    r = np.empty(K.shape, dtype=np.ndarray)
    for i,j,k in np.ndindex((N,N,N)):
            K[i,j,k] = l(i,N)*b1 + l(j,N)*b2 + l(k,N)*b3
            r[i,j,k] = (1/N)*(i*a1 + j*a2 + k*a3)
    return K,r
def generate_potential(K):
    return vector_U(K)
def get_ek(K,k):
    q=K+k
    return c * q.dot(q)
vector_get_ek = np.vectorize(get_ek,excluded=[1]) # only vectorize zeroth argument
def get_H_block(k,K,U_K):
    # following Marder's Table 7.1
    # eigenvalues of a block-diagonal matrix are simply the eigenvalues of each block
    # can therefore compute only the block for this k
    H = np.diagflat(vector_get_ek(K,k))
    m,n = np.indices(H.shape)
    H += U_K.flat[np.abs(m-n)]
    return H
# take advantage of only the diagonal changing under a change of k
def efficient_new_H(H,k,K,U_K):
    # I measure a speedup of order N over get_H_block
    np.fill_diagonal(H,vector_get_ek(K,k).flat + U_K[0,0,0])
    return H
bands=6
# high symmetry points for the reciprocal lattice
L     = 2*np.pi/a*np.array([1/2,1/2,1/2])
gamma = 2*np.pi/a*np.array([  0,  0,  0])
X     = 2*np.pi/a*np.array([  0,  1,  0])
pointU= 2*np.pi/a*np.array([1/4,  1,1/4])

path = [L, gamma, X, pointU, gamma]
labels = ['L', r'$\Gamma$', 'X', 'U', r'$\Gamma$']
ksteps = 50
N = 12
energies = np.zeros([ksteps*(len(path)-1),bands])
kscale = np.zeros(ksteps*(len(path)-1))
K,r = generate_lattice(N)
U_K = generate_potential(K)

label_locations = []
H = get_H_block(np.array([0.,0.,0.]),K,U_K)

for i in range(0,len(path)-1):
    label_locations.append(i*ksteps)
    kstep = (path[i+1]-path[i]) / ksteps
    for step in range(ksteps):
        qval = path[i]+step*kstep
        H = efficient_new_H(H,qval,K,U_K)
        energies[step+i*ksteps,:] = LA.eigvalsh(H,eigvals=(0,bands-1))
        kscale[step+i*ksteps] = LA.norm(kstep)

label_locations.append(step+i*ksteps)
kvals = np.cumsum(kscale) # do this to scale the x-axis linearly with k-distance covered

font = {'family' : 'serif', 'weight' : 'bold', 'size' : 18}
matplotlib.rc('text', usetex=True)
matplotlib.rcParams['text.latex.preamble']=[r"\boldmath"]
matplotlib.rc('font', **font)
matplotlib.ticker.ScalarFormatter(useMathText=True)

plt.ylabel('band energy (eV)')
plt.xticks([kvals[x] for x in label_locations], labels)
plt.axes().xaxis.grid(True,which='major')
plt.axes().tick_params(pad=18)
for n in range(bands):
    plt.plot(kvals,energies[:,n])
plt.savefig('al_bands.pdf',bbox_inches='tight')
```

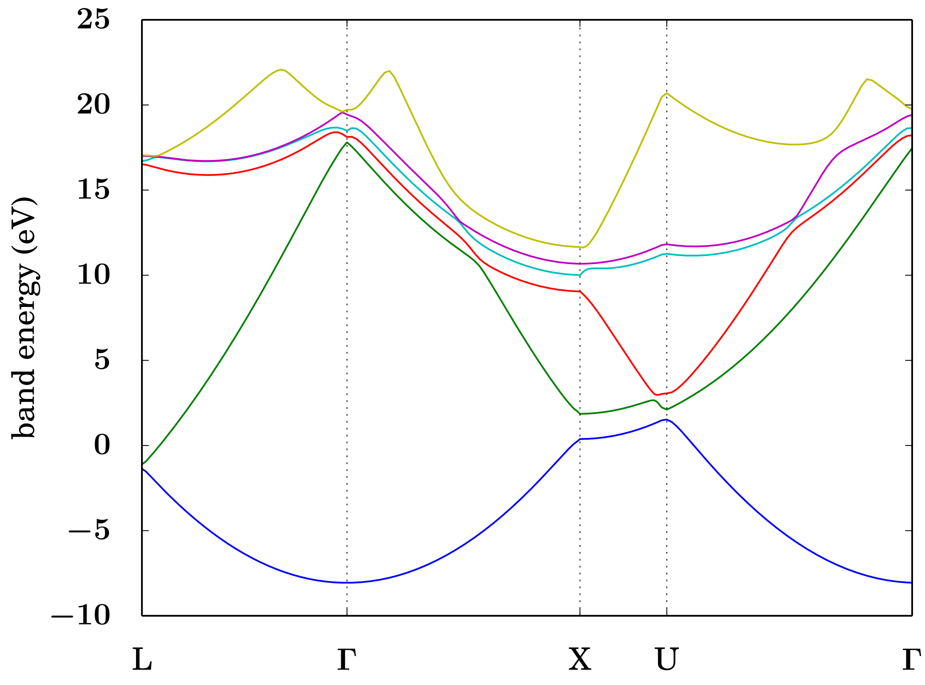

Figure 3: The lowest six bands produced by my code for Aluminum, truncating the reciprocal lattice after $N = 12$ values for each basis vector ($12^3 = 1728$ reciprocal lattice sites). From the potential $U(\vec{k} = 0) \approx -7.8$ eV a Fermi energy $\varepsilon_F = -\frac{3}{2}U(0) \approx 11.7$ eV is calculated. This value is consistent with reference data [1].

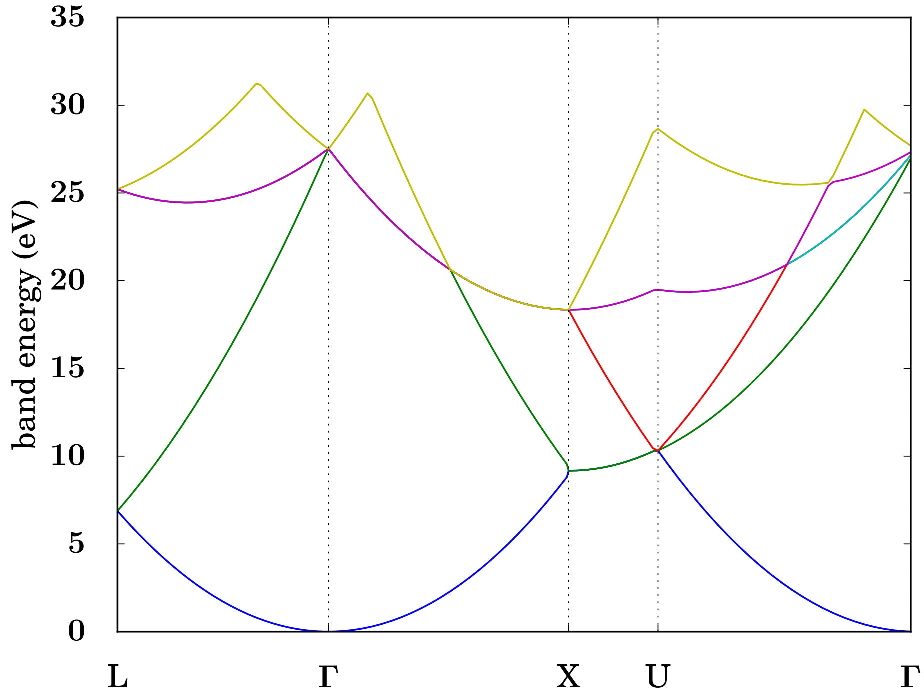

Figure 4: The lowest six bands produced by my code for the free electron in an fcc lattice, truncating the reciprocal lattice after $N = 12$ values for each basis vector. This was achieved by modifying the potential in the above code `def U(K): return 0`.

References

[1] N.W. Ashcroft and N.D. Mermin. *Solid State Physics*. Saunders College, Philadelphia, 1976.

[2] The Scipy community. Numpy reference – numpy v1.9 manual. `http://docs.scipy.org/doc/numpy/reference/`. Accessed: 2015-05-10.

[3] Yuan Ming Huang. Lecture notes on solid state physics. `http://www.lcst-cn.org/Solid%20State%20Physics/Ch25.html`. Accessed: 2015-05-10.

[4] Michael P. Marder. *Condensed Matter Physics*. Wiley, 2nd edition, 2011.