# Report for free electron model for metal films

BY JINLONG HUANG

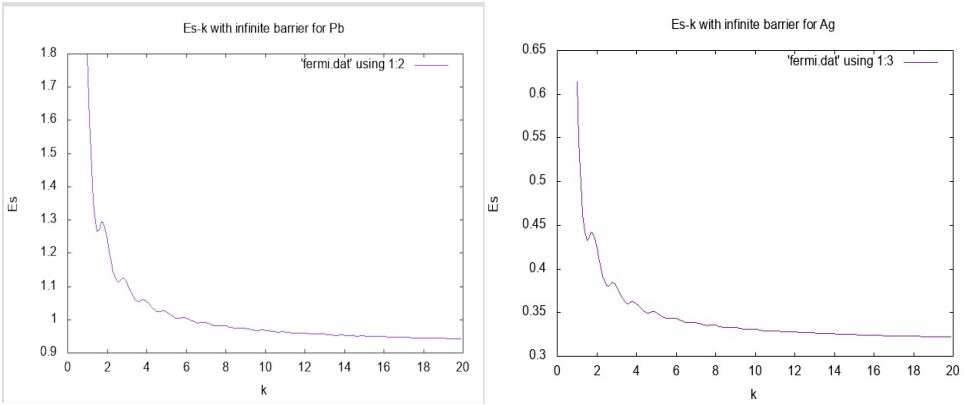
Dec 31, 2016

This project plots the film surface energy $Es$ as a function of $\kappa$ for Pb(111) and Ag(111) films in infinite barrier and finite barrier. The formulas are calculated from free electron model and consider the restriction by metal film in z direction.

Two programs fermi.f90 and fermiFinite.f90 deal with the infinite situation and finite situation respectively. I'll introduce fermi.f90 firstly.

1. Infinite barrier

   The file fermi.f90 firstly calculates the principle quantum number N for fermi energy, using the inequality $E_N \leqslant E_f < E_{N+1}$, where $E_f = E_F\left[\frac{2\kappa}{3N} + \frac{(N+1)(2N+1)}{6\kappa^2}\right]$ and $E_n = \frac{n^2}{\kappa^2} E_F$ . Then it computes $E_s = \frac{k_F^2 E_F}{4\pi}\kappa\left\{S - \frac{4}{5}\right\}$, where $S = \frac{4\kappa}{9N} + \frac{2(N+1)(2N+1)}{9\kappa^2} - \frac{N(N+1)(2N+1)(8N^2+3N-11)}{180\kappa^5}$, and $k_F$ , $E_F$ are constant for Pb and Ag. At last, it output k, EsPb, EsAg into file fermi.dat, which will be used to plot Es-k for Pb and Ag using gnuplot. The graph is showed in files Pb_infinite.png and Ag_infinite.png.




   As we can see from the graph, both surface energy decrease as k increase. The range for Pb is from 0.9 to 1.8, which is larger than Ag.

   The infinite barrier is just a theoretical assumption to easily calculate the surface energy. For more realistic situation, we need to consider when the barrier is finite.

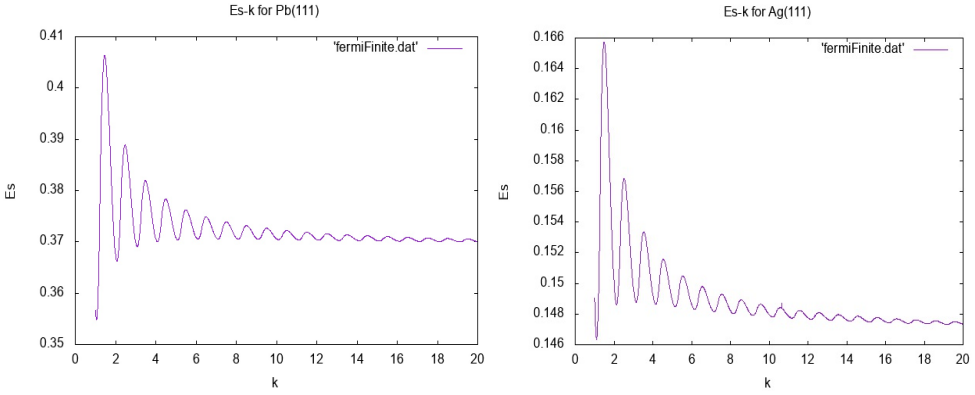
2. Finite barrier

   The file fermiFinite.f90 firstly calculate the result for Pb(111), using barrier $\frac{V_0}{E_F} = 1.45$. This file is more complicated than fermi.f90, with two function fPb and yita_N called many times in the program. Function fPb takes inputs (yita, k, N, V0Pb), then computes the function $f = \eta_n - n + \frac{2}{\pi}\sin^{-1}\left(\frac{\eta_n}{\kappa\sqrt{V_0/E_F}}\right)$. Function fPb is only called in function yita_N. Function yita_N takes inputs (k, N, dl, V0Pb), then computing the zero point of $f$ , which equals $\eta_n$ that we wanted. I used bisection method in yita_N to calculate the zero point, which is simple and intuitive.

   The main program can be divided into two parts. The first part the principle number for fermi energy as in program fermi.f90. The second part calculates the surface energy and output the results into file fermiFinite.dat. The difference to fermi.f90 is that fermiFinite.f90 uses different formulas to both N and $Es$.

To compute N, I firstly called yita_N to calculate $E_N$ and $E_{N+1}$. Then I compute $E_f(\kappa) = E_F\left[\frac{2\kappa}{3N} + \frac{1}{N}\sum_{n\leqslant N}\frac{E_n}{E_F}\right]$, where $E_n = \frac{\eta_n^2}{\kappa^2}E_F$. After this I iterate N from 1 to satisfy condition $E_N \leqslant E_f < E_{N+1}$. To this step we get the N for fermi energy. In second part I compute $E_s = \frac{k_F^2 E_F}{4\pi}\left\{\sum_{n\leqslant N}\frac{E_f^2 - E_n^2}{E_F^2} - \frac{4\kappa}{5}\right\}$ and output k, $E_s$ to file fermiFinite.dat.

These are the graph I get:




The step for k is 0.01, which means in each graph there are 2000 points. $E_s$ for Pb(111) bounced between 0.35 and 0.41, and Es for Ag(111) bounced between 0.14 and 0.16. These results are perfectly consistent with Figure 15. and 16. in the paper:

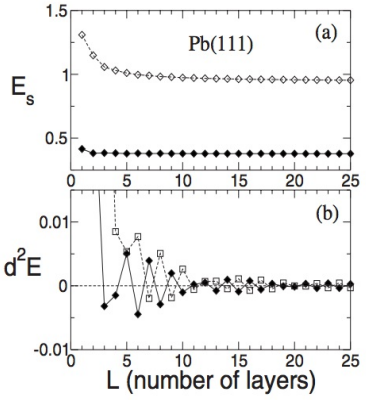



FIG. 16. Interface energy $E_s$ and its second difference for a freestanding Pb(111) thin film. For Pb, $W_m$=4.25 eV and $V_0$ =1.45$E_F$. The open diamonds with dashed lines are for infinite barriers. The unit of $E_s$ is eV/Å$^2$.

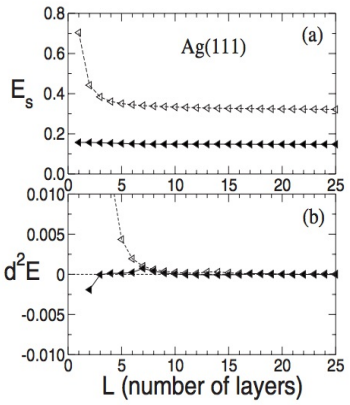



FIG. 15. Interface energy $E_s$ and its second difference for a freestanding Ag(111) thin film. For Ag, $W_m$=4.3 eV and $V_0$ =1.78$E_F$. The open triangles with dashed lines are for infinite barriers. The unit of $E_s$ is eV/Å$^2$.

During the development of fermiFinite.f90, I learnt a lot about how to code accurately to achieve the function that I want.

a. Carefully check the expression. Although this point was so obvious, when I developed the program, I did make many similar mistakes, like placing parentheses in wrong position, mistaking $\leqslant$ for $<$, losing square and coefficient et. al. This mistake are very easily to make, and at the same time very hard to debug. So it's very necessary to give special attention on the accuracy of expression.

b. Testing not debugging. This motto can be actually applied in all kinds of programming. Testing means when we develop a program, we should add test modulo (for example, print) in every major part of the program rather than after writing the whole program to debug. This kind of debug can be very time-comsumming, since the range to check can be very large.

c. Define all variables and functions and use them in the right way. Check the definition and the places where the functions are called.

d. In loop, initialize some necessary variables and increase the iterative number. These mistakes can be very hard to find.

e. Carefully choose the precision. When I choose the precision in bisection method to be 1.0e-6, I can only calculate k from 1.0 to 15.0. After testing using print statement, I adjusted the precision to 5.0e-6, which enable me to compute k from 1.0 to 20.0.

3. After this project, I feel happy that I have more familarity with fortran and have more confidence to programming using fortran.