Extension of Kronig-Penney model to metal alloy like perovskites

Domenico Oricchio

June 14, 2025

Abstract

I generalize the Kronig-Penney model for multiple barrier potentials of different values. I write a different way to solve the rectangular potential barrier so to simplify the solutions. I try to understand the complexity of energy band in the alloys to optimize the photons absorption using a random modification of model parameters.

The model of Kronig-Penney is a quantum system with an infinite periodic array of rectangular potentials, and the model use the Bloch's theorem to connect the wavefunction to the boundaries of the potentials.

The problem of Kronig-Penney model is the complexity of the solutions, that makes it almost impossible to deal with models with different barrier potentials[1].

I simplify the solution using the Laplace transform for the Schrödinger equation, that incorporate the initial condition in the transform of the derivative; so that it is simple to reiterate the transform to different potential barrier.

I expect a band structure that optimize the photon absorption, with a complex band gap because of there are two, or more, different atoms in the semiconductor that give different solutions of the electron interaction with the atoms.

I study the a semiconductor with two different atoms, but the theory is simply generalizable; I use the Laplace transform for the generic U barrier:

$$
\begin{aligned}
&-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\Psi(x) + U(x)\Psi(x) = E\Psi(x) \\
&\mathcal{L}\left\{-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\Psi(x) + U(x)\Psi(x)\right\} = \mathcal{L}\left\{E\Psi(x)\right\} \\
&-\frac{\hbar^2}{2m}\left[s^2\mathcal{L}\{\Psi(x)\} - s\Psi(0) - \frac{d\Psi(0)}{dx}\right] = (E-U)\mathcal{L}\left\{\Psi(x)\right\}
\end{aligned}
$$

[1] It is possible to use computer programs, but the propierty of the solutions are not evident

in the alloy there are three potentials U_3, U_1 and $U_2 = U_0 = 0$, and I can write the generic solution for U:

$$\left[s^2 + \frac{2m(E-U)}{\hbar^2}\right]\mathcal{L}\{\Psi(x)\} = [s^2 + K^2]\,\mathcal{L}\{\Psi(x)\} = s\Psi(0) + \frac{d\Psi(0)}{dx}$$
$$\mathcal{L}\{\Psi(x)\} = \frac{s\Psi(0) + \dfrac{d\Psi(0)}{dx}}{s^2 + K^2}$$

the generic solution of the Laplace transform is:

$$\Psi(x) = \cos(Kx)\Psi(0) + \frac{\sin(Kx)}{K}\Psi'(0)$$

this solution is true for each boundary of the potential barrier, and I can write the solution for a generic boundary, for example the x_i boundary:

$$\Psi(x) = \cos(K(x-x_i))\Psi(0) + \frac{\sin(K(x-x_i))}{K}\Psi'(0)$$
$$\Psi'(x) = -\sin(K(x-x_i))\Psi(0) + \cos(K(x-x_i))\Psi'(0)$$

this can be write like a transformation matrix for $x_i \le x \le x_{i+1}$:

$$\begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} = \begin{pmatrix} \cos(K_i(x-x_i)) & \dfrac{\sin(K_i(x-x_i))}{K_i} \\ -K_i\sin(K_i(x-x_i)) & \cos(K_i(x-x_i)) \end{pmatrix} \begin{pmatrix} \Psi(x_i) \\ \Psi'(x_i) \end{pmatrix} =$$
$$= \begin{pmatrix} \cos(K_i(x-x_i)) & (x-x_i)sinc(K_i(x-x_i)) \\ -K_1\sin(K_i(x-x_i)) & \cos(K_i(x-x_i)) \end{pmatrix} \begin{pmatrix} \Psi(x_i) \\ \Psi'(x_i) \end{pmatrix}$$

the function in the matrix can have complex value, because of $K_i = \sqrt{\dfrac{2m}{\hbar^2}(E-U_i)}$ and it is possible that $E < U_i$; I use the complex value of the trigonometric functions:

$$\cos(i\alpha) = \cosh(\alpha)$$
$$\sin(i\alpha) = i\sinh(\alpha)$$
$$sinc(i\alpha) = \frac{\sin(i\alpha)}{i\alpha} = sinch(\alpha)$$

The wavefunction on the generic boundary is obtained applying N-times the transformation matrix:

$$\begin{pmatrix} \Psi(x_{i+1}) \\ \Psi'(x_{i+1}) \end{pmatrix} = \prod_{i=1}^{N} \begin{pmatrix} \cos(K_{N-i}\Delta_{N-1}) & \Delta_{N-i}\,sinc(K_{N-i}\Delta_{N-i}) \\ -K_{N-i}\,\sin(K_{N-i}\Delta_{N-i}) & \cos(K_{N-i}\Delta_{N-i}) \end{pmatrix} \begin{pmatrix} \Psi(x_0) \\ \Psi'(x_0) \end{pmatrix}$$

the transformation matrix has the property:

$$\det\begin{pmatrix} \cos(K_i\Delta_i) & \dfrac{\sin(K_i\Delta_i)}{K_i} \\ -K_i\sin(K_i\Delta_i) & \cos(K_i\Delta_i) \end{pmatrix} = 1$$

so that each product of transformation matrices has determinant equal to one, so that the area of the surface is preserved.

The rectangular potentials are periodicals, so thath the Bloch's theorem can be used:

$$\begin{pmatrix} \Psi_k(x+\Delta_0+\Delta_1) \\ \Psi_k'(x+\Delta_0+\Delta_1) \end{pmatrix} = \begin{pmatrix} e^{ik(\Delta_0+\Delta_1)} & 0 \\ 0 & e^{ik(\Delta_0+\Delta_1)} \end{pmatrix} \begin{pmatrix} \Psi_k(x) \\ \Psi_k'(x) \end{pmatrix}$$

so that I can equal the wavefunction with the N Laplace transform and the Bloch's value in the last boundary:

$$\begin{pmatrix} \cos(K_1\Delta_1) & \Delta_1 sinc(K_1\Delta_1) \\ -K_1\sin(K_1\Delta_1) & \cos(K_1\Delta_1) \end{pmatrix} \begin{pmatrix} \cos(K_0\Delta_0) & \Delta_0 sinc(K_0\Delta_0) \\ -K_0\sin(K_0\Delta_0) & \cos(K_0\Delta_0) \end{pmatrix} \begin{pmatrix} \Psi_k(x) \\ \Psi_k'(x) \end{pmatrix} =$$
$$= \begin{pmatrix} e^{ik(\Delta_0+\Delta_1)} & 0 \\ 0 & e^{ik(\Delta_0+\Delta_1)} \end{pmatrix} \begin{pmatrix} \Psi_k(x) \\ \Psi_k'(x) \end{pmatrix}$$

the non trivial solution of the problem happen when

$$\det\left(\begin{pmatrix} \cos(K_1\Delta_1) & \Delta_1 sinc(K_1\Delta_1) \\ -K_1\sin(K_1\Delta_1) & \cos(K_1\Delta_1) \end{pmatrix} \begin{pmatrix} \cos(K_0\Delta_0) & \Delta_0 sinc(K_0\Delta_0) \\ -K_0\sin(K_0\Delta_0) & \cos(K_0\Delta_0) \end{pmatrix} - \begin{pmatrix} e^{ik(\Delta_0+\Delta_1)} & 0 \\ 0 & e^{ik(\Delta_0+\Delta_1)} \end{pmatrix}\right) = 0$$

I write the transformation matrix like $\mathbf{\Lambda}$, and the equation can be write like:

$$0 = \det\left(\mathbf{\Lambda} - \begin{pmatrix} e^{ik(\Delta_0+\Delta_1)} & 0 \\ 0 & e^{ik(\Delta_0+\Delta_1)} \end{pmatrix}\right) = \det\left(\begin{pmatrix} \Lambda_{11} - e^{ik(\Delta_0+\Delta_1)} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} - e^{ik(\Delta_0+\Delta_1)} \end{pmatrix}\right) =$$
$$= \left(\Lambda_{11} - e^{ik(\Delta_0+\Delta_1)}\right)\left(\Lambda_{22} - e^{ik(\Delta_0+\Delta_1)}\right) - \Lambda_{12}\Lambda_{21}$$

if $\alpha = k(\Delta_0+\Delta_1)$

$$0 = \Lambda_{11}\Lambda_{22} - \Lambda_{12}\Lambda_{21} - \Lambda_{11}e^{i\alpha} - \Lambda_{22}e^{i\alpha} + e^{2i\alpha} = 1 - \Lambda_{11}[\cos(\alpha) + i\sin(\alpha)] - \Lambda_{22}[\cos(\alpha) + i\sin(\alpha)] +$$
$$+ \cos(2\alpha) + i\sin(2\alpha) = 1 - \Lambda_{11}\cos(\alpha) - \Lambda_{22}\cos(\alpha) + 2\cos^2(\alpha) - 1 - i\Lambda_{11}\sin(\alpha) - i\Lambda_{22}\sin(\alpha) +$$
$$+2i\sin(\alpha)\cos(\alpha) = \cos(\alpha)[-\Lambda_{11} - \Lambda_{22} + 2\cos(\alpha)] + i\sin(\alpha)[-\Lambda_{11} - \Lambda_{22} + 2\cos(\alpha)]$$

the determinant is null if

$$\Lambda_{11} + \Lambda_{22} = 2\cos(\alpha)$$
$$\boxed{Trace(\Lambda) = 2\cos(\alpha)}$$

it is simple to obtain the trace of a multiplication of transformation of matrices:

$$Trace\left(\prod_{i=1}^{N}\mathbf{\Lambda}^{(i)}\right)=\sum_{k_1,\cdots,k_N}\Lambda_{k_1k_2}^{(i_1)}\Lambda_{k_2k_3}^{(i_2)}\cdots\Lambda_{k_Nk_1}^{(i_n)}$$

so that the number of terms in the trace is equal to 2^N-1.

The terms for the usual Kronig-Penney model with a single potential barrier is:

$$\cos(K_1\Delta_1)\cos(K_0\Delta_0)-\Delta_1 sinc(K_1\Delta_1)K_0\sin(K_0\Delta_0)-K_1\sin(K_1\Delta_1)\Delta_0 sinc(K_0\Delta_0)+\cos(K_1\Delta_1)\cos(K_0\Delta_0)=$$
$$=\boxed{2\cos(K_1\Delta_1)\cos(K_0\Delta_0)-\Delta_1 sinc(K_1\Delta_1)K_0\sin(K_0\Delta_0)-K_1\sin(K_1\Delta_1)\Delta_0 sinc(K_0\Delta_0)=2\cos\left(k(\Delta_0+\Delta_1)\right)}$$

there are two regions for the k solutions (each k satisfy the equation if the first member has values between -2 and 2); in the first member I have:

$$K_0=\sqrt{\frac{2m}{\hbar^2}E}$$
$$K_1=\sqrt{\frac{2m}{\hbar^2}(E-U)}$$

if $E<U$ then $K_1=i|K_1|$:

$$\boxed{2\cosh(|K_1|\Delta_1)\cos(K_0\Delta_0)-\Delta_1 sinch(|K_1|\Delta_1)K_0\sin(K_0\Delta_0)+|K_1|\sinh(|K_1|\Delta_1)\Delta_0 sinc(K_0\Delta_0)=2\cos\left(k(\Delta_0+\Delta_1)\right)}$$

that it is the classical result of Kronig-Penney.

I write the solution for two potential barriers U_3 and U_1, with two free particles regions (null barriers $U_2=U_0=0$).

The transformation matrix is the product of four matrix with determinant one:

$$\begin{pmatrix}\cos(K_i\Delta_i) & \Delta_i sinc(K_i\Delta_i)\\ -K_i\sin(K_i\Delta_i) & \cos(K_i\Delta_i)\end{pmatrix}$$

and the trace of the final matrix is simply the sum of $2^4 - 1 = 15$ terms, that I write:

$$\begin{aligned}
&2\cos(K_3\Delta_3) && \cos(K_2\Delta_2) && \cos(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && \cos(K_2\Delta_2) && \Delta_1 sinc(K_1\Delta_1) && K_0\sin(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && \Delta_2 sinc(K_2\Delta_2) && K3_1\sin(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && \Delta_2 sinc(K_2\Delta_2) && \cos(K_1\Delta_1) && K_0\sin(K_0\Delta_0)+\\
&-\Delta_3 sinc(K_3\Delta_3) && K_2\sin(K_2\Delta_2) && \cos(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&+\Delta_3 sinc(K_3\Delta_3) && K_2\sin(K_2\Delta_2) && \Delta_1 sinc(K_1\Delta_1) && K_0\sin(K_0\Delta_0)+\\
&-\Delta_3 sinc(K_3\Delta_3) && \cos(K_2\Delta_2) && K_1\sin(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&-\Delta_3 sinc(K_3\Delta_3) && \cos(K_2\Delta_2) && \cos(K_1\Delta_1) && K_0\sin(K_0\Delta_0)+\\
&-K_3\sin(K_3\Delta_3) && \cos(K_2\Delta_2) && \cos(K_1\Delta_1) && \Delta_0 sinc(K_0\Delta_0)+\\
&-K_3\sin(K_3\Delta_3) && \cos(K_2\Delta_2) && \Delta_1 sinc(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&+K_3\sin(K_3\Delta_3) && \Delta_2 sinc(K_2\Delta_2) && K_1\sin(K_1\Delta_1) && \Delta_0 sinc(K_0\Delta_0)+\\
&-K_3\sin(K_3\Delta_3) && \Delta_2 sinc(K_2\Delta_2) && \cos(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && K_2\sin(K_2\Delta_2) && \cos(K_1\Delta_1) && \Delta_0 sinc(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && K_2\sin(K_2\Delta_2) && \Delta_1 sinc(K_1\Delta_1) && \cos(K_0\Delta_0)+\\
&-\cos(K_3\Delta_3) && \cos(K_2\Delta_2) && K_1\sin(K_1\Delta_1) && \Delta_0 sinc(K_0\Delta_0) =\\
&= 2cos(k(\Delta_3+\Delta_2+\Delta_1+\Delta_0))
\end{aligned}$$

the $K_0(E), K_1(E), K_2(E), K_3(E)$ are function of the energy of the free particle; and there are three equation for the extended Kronig-Penney equation (if the barrier potential are in an ordered sequence), that correspond to the three complex-value changes in the K function (I suppose that $U_1 < U_3$):

$$\begin{aligned}
E \le U_1 &\implies K_1(E) = i|K_1(E)|,\ K_3(E) = i|K_3(E)|\\
U_1 \le E \le U_3 &\implies K_1(E) = |K_1(E)|,\ K_3(E) = i|K_3(E)|\\
U_3 \le E &\implies K_1(E) = |K_1(E)|,\ K_3(E) = |K_3(E)|
\end{aligned}$$

when the free energy is great, then the sinc function tends to zero, so that only the first term is non null, and the first term is included in $[-2, 2]$, so that the energy gap tend to zero.

The energy gap happen in the bounded state, where the K function have complex variable, and if $U_1 < U_3$ I want that the trigonometric functions with real value have a variation greater of 2π, so that there is a not simple dispersion relation: if the solutions of the equation for k values is not simple, then the Fermi surface of the electron is not simple and there are many electrons near the Fermi surface that can absorb the photons without phonon absorption, with a direct transition to the near free band, and there are many maximums in the Fermi surface, so that the transition between the maximum of the Fermi surface and the conduction band edge is greater of the pure semiconductor.

The $K_1(E)$ electron remain a free particle solution until the U_3 value, and $K_3(E)$ is a bounded electron solution, so that a not simple solution is:

$$\sqrt{K_1(U_3)\Delta_1} = \sqrt{\frac{2m}{\hbar^2}(U_3 - U_1)\Delta_1} \approx 2\pi$$

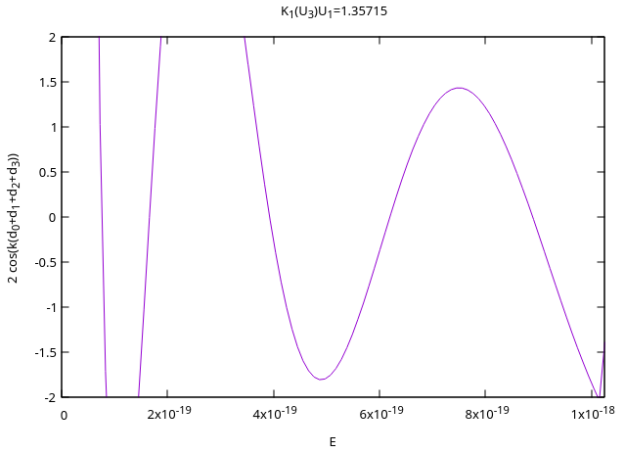
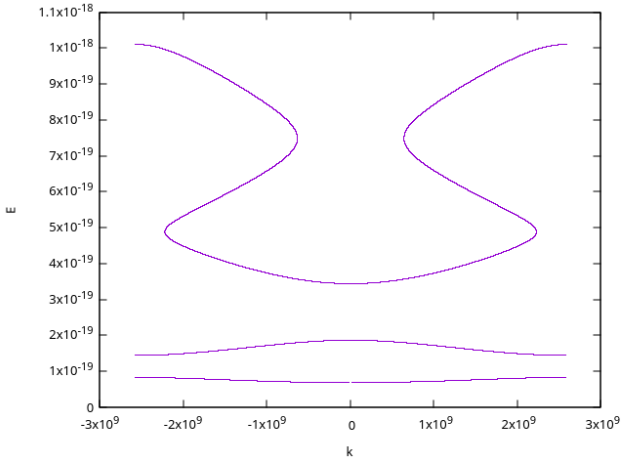


Figure 1: A solution of the Kronig-Penney model with $\Delta_0 = 3.2760749217856\mathring{A}; \Delta_1 = 1.08244888593528\mathring{A}; \Delta_2 = 2.8470487432461\mathring{A}; \Delta_3 = 4.94205910669895\mathring{A}; U_1 = 6.39860025918eV; U_3 = 0.409689081618919677 74eV;$



¿

Figure 2: Energy gaps for the Figure 1 Kronig-Penney model

that it is the condition that a semiconductor must satisfy to increase the photon absorption.

The theory is generalizable to more component alloys, with a expected increase of complexity of the Fermi surface.

In general if there are N potential barrier, in ascending order, $U_1 < U_3 < \cdots < U_N$ then the condition for a not simple condition is that:

$$\exists i \ni \sqrt{K_i(U_{i+2})\Delta_i} = \sqrt{\frac{2m}{\hbar^2}(U_{i+2} - U_i)\Delta_i} \approx 2\pi$$

I calculate the trace with the help of C program, and I calculate the dispersion relation in the range $[0 : U_3]$ with a C program, and the results are shown in the figures in the article.

C language program

```
/*************************************************************************
*                          Kronig_Penney.c                               *
*                         Copyright (C) 2018                             *
*     Oricchio Domenico - Theorical Physicist from Salerno University    *
*                                                                        *
*  This program is free software: you can redistribute it and/or modify  *
*  it under the terms of the GNU General Public License as published by  *
*  the Free Software Foundation, either version 3 of the License, or     *
*  (at your option) any later version.                                   *
*                                                                        *
*  This program is distributed in the hope that it will be useful,       *
*  but WITHOUT ANY WARRANTY; without even the implied warranty of        *
*  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
*  GNU General Public License for more details.                          *
*                                                                        *
*  You should have received a copy of the GNU General Public License     *
*  along with this program.  If not, see <https://www.gnu.org/licenses/>. *
*************************************************************************/
/** HEADER FILES di ogni programma standard C */
#define _GNU_SOURCE
#include <sched.h>
#include <ctype.h>
#include <errno.h>
#include <float.h>
#include <fcntl.h>
#include<limits.h>
#include<locale.h>
#include  <math.h>
#include<setjmp.h>
#include<signal.h>
#include<stdarg.h>
#include<stddef.h>
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
#include  <time.h>
#include<unistd.h>

double sinc(double x);

double sinch(double x);

int main( )
{
 int     n,
 N=1000000,
 j;
 double h = 1.054571817e-34,
 m = 9.1093837e-31,
 C = 2.0*m/(h*h),
```

```
E,
dE,
d[4],
U[4],
K[4],
k,
R=0.0;
FILE *file_name;

file_name = fopen( "./file.swp", "w" );

d[0] = 3.2760749217856e-10; d[1] = 1.08244888593528e-10; d[2] = 2.8470487432461e-10; d[3] = 4.94205910669895e-10; U[1] = 1.02516878255589e-18; U[3] = 6.56394273774752e-20;

if( U[3]>U[1] ){ dE=U[3]/(double)N; }
else           { dE=U[1]/(double)N; }

for( n=0, E=U[2]=U[0]=0.0; n<N; n++ )
{
 for( j=0; j<=3; j++ ){ K[j] = sqrt( C*fabs(E-U[j])); }

 if( E>=U[1] && E>=U[3] )
 {
  R =      cos(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinc(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  -        cos(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sin(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cos(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - d[3]* sinc(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  + d[3]* sinc(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinc(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - d[3]* sinc(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sin(K[1]*d[1])*        cos(K[0]*d[0])
  - d[3]* sinc(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinc(K[1]*d[1])*        cos(K[0]*d[0])
  + K[3]*  sin(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sin(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cos(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  -        cos(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinc(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sin(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  +        cos(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0]);
 }
 if( E>=U[3] && E<=U[1] )
 {
  R =      cos(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinch(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  +        cos(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cosh(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - d[3]* sinc(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
  + d[3]* sinc(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinch(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  + d[3]* sinc(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])*        cos(K[0]*d[0])
  - d[3]* sinc(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinch(K[1]*d[1])*        cos(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  - K[3]*  sin(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
  -        cos(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cosh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  -        cos(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinch(K[1]*d[1])*        cos(K[0]*d[0])
  +        cos(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  -        cos(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0]);
 }
 if( E<=U[3] && E>=U[1] )
 {
  R =      cosh(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  -        cosh(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinc(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  -        cosh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sin(K[1]*d[1])*        cos(K[0]*d[0])
  -        cosh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cos(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - d[3]* sinch(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  + d[3]* sinch(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinc(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  - d[3]* sinch(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sin(K[1]*d[1])*        cos(K[0]*d[0])
  - d[3]* sinch(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
  + K[3]*  sinh(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  + K[3]*  sinh(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinc(K[1]*d[1])*        cos(K[0]*d[0])
  - K[3]*  sinh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sin(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  + K[3]*  sinh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0])
  -        cosh(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cos(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  -        cosh(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinc(K[1]*d[1])*        cos(K[0]*d[0])
  -        cosh(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sin(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
  +        cosh(K[3]*d[3])*        cos(K[2]*d[2])*        cos(K[1]*d[1])*        cos(K[0]*d[0]);
 }
```

```
  if( E<=U[3] && E<=U[1] )
  {602176634
   R =      cosh(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
   -        cosh(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinch(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
   +        cosh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])*        cos(K[0]*d[0])
   -        cosh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cosh(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
   - d[3]* sinch(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
   + d[3]* sinch(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinch(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
   + d[3]* sinch(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])*        cos(K[0]*d[0])
   - d[3]* sinch(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])* K[0]*  sin(K[0]*d[0])
   + K[3]*  sinh(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
   + K[3]*  sinh(K[3]*d[3])*        cos(K[2]*d[2])* d[1]* sinch(K[1]*d[1])*        cos(K[0]*d[0])
   + K[3]*  sinh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
   + K[3]*  sinh(K[3]*d[3])* d[2]* sinc(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0])
   -        cosh(K[3]*d[3])* K[2]*  sin(K[2]*d[2])*        cosh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
   -        cosh(K[3]*d[3])* K[2]*  sin(K[2]*d[2])* d[1]* sinch(K[1]*d[1])*        cos(K[0]*d[0])
   +        cosh(K[3]*d[3])*        cos(K[2]*d[2])* K[1]*  sinh(K[1]*d[1])* d[0]* sinc(K[0]*d[0])
   +        cosh(K[3]*d[3])*        cos(K[2]*d[2])*        cosh(K[1]*d[1])*        cos(K[0]*d[0]);
  }
  if( fabs(R)<=2.0 )
  {
   k = acos( R*0.5)/ (d[0]+d[1]+d[2]+d[3] );
   printf( "\rk=% e E=% e R=% e    ", k, E, R );
   // fprintf( file_name, "% e % e\n", E, k );
   fprintf( file_name, "% e % e\n", k, E );
   fprintf( file_name, "% e % e\n", -k, E );
  }
  E += dE;
 }
 printf( "\n" );
 fclose( file_name );
 exit( EXIT_SUCCESS);
 return( EXIT_SUCCESS);
}

double sinc(double x)
{
 double val;

 if( fabs(x)<=1e-9 ){ val=1.0-x*x/6.0; }
 else               { val=sin(x)/x; }

 return val;
}

double sinch(double x)
{
 double val;

 if( fabs(x)<=1e-9 ){ val=1.0+x*x/6.0; }
 else               { val=sinh(x)/x; }

 return val;
}
```

References

[1] Ashcroft, N. W., Mermin, N. D. (1976). Solid State Physics. Holt-Saunders.

[2] Kittel, C. (2004). Introduction to solid state physics (8th ed.). John Wiley & Sons.

[3] Gratias, D.. (2012). The adventure of quasicrystals: A sucessful multidisciplinary effort. Europhysics News. 43. 26-29. 10.1051/epn/2012502.