**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий. механики и оптики Изображение выглядит как текст, коллекция картинок, посуда

Автоматически созданное описание** **УЧЕБНЫЙ ЦЕНТР ОБЩЕЙ ФИЗИКИ ФТФ**

Группа М32111 К работе допущен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Студент Соловьев Е.С. Писарев Е.С. Акберов Р.Х. Работа выполнена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

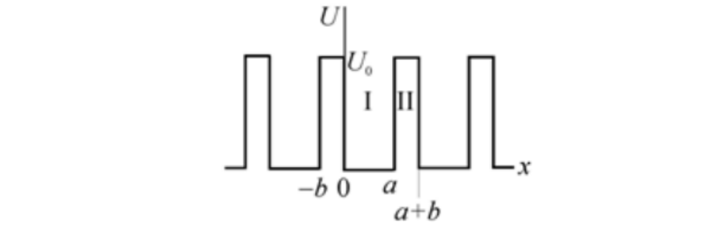
Преподаватель Шоев В.И. Отчет принят\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рабочий протокол и отчет по моделированию №2

**Зонная структура одномерного кристалла**

1. Теория:

В модели почти свободных электронов, которую предложили Крониг и Пени, рассматривается движение электрона в линейной цепочке прямоугольных потенциальных ям. Ширина ям равна α и они отделены друг от друга потенциальными барьерами толщиной b и высотой U0. Длина цепочки равна L, а период цепочки равен c=a+b



Пусть E – энергия электрона. Состояние электрона описывается уравнением Шредингера:

Решение для области I:

Первое слагаемое соответствует прямой волне, а второе – волне, отраженной от барьера.

Решение для области II:

Где коэффициенты:

Вместо ψ1 и ψ2 подставим одномерную функцию Блоха:

Определим A, B, C, D используя склейку:

Придем к уравнению:

Рассмотрим предельный случай, устремив ширину барьера b к нулю и увеличим высоту барьера, устремив ее к бесконечности (). При этом потребуем, чтобы произведение bU0 оставалось постоянным.

Обозначим, где P – величина, характеризующая прозрачность барьера.

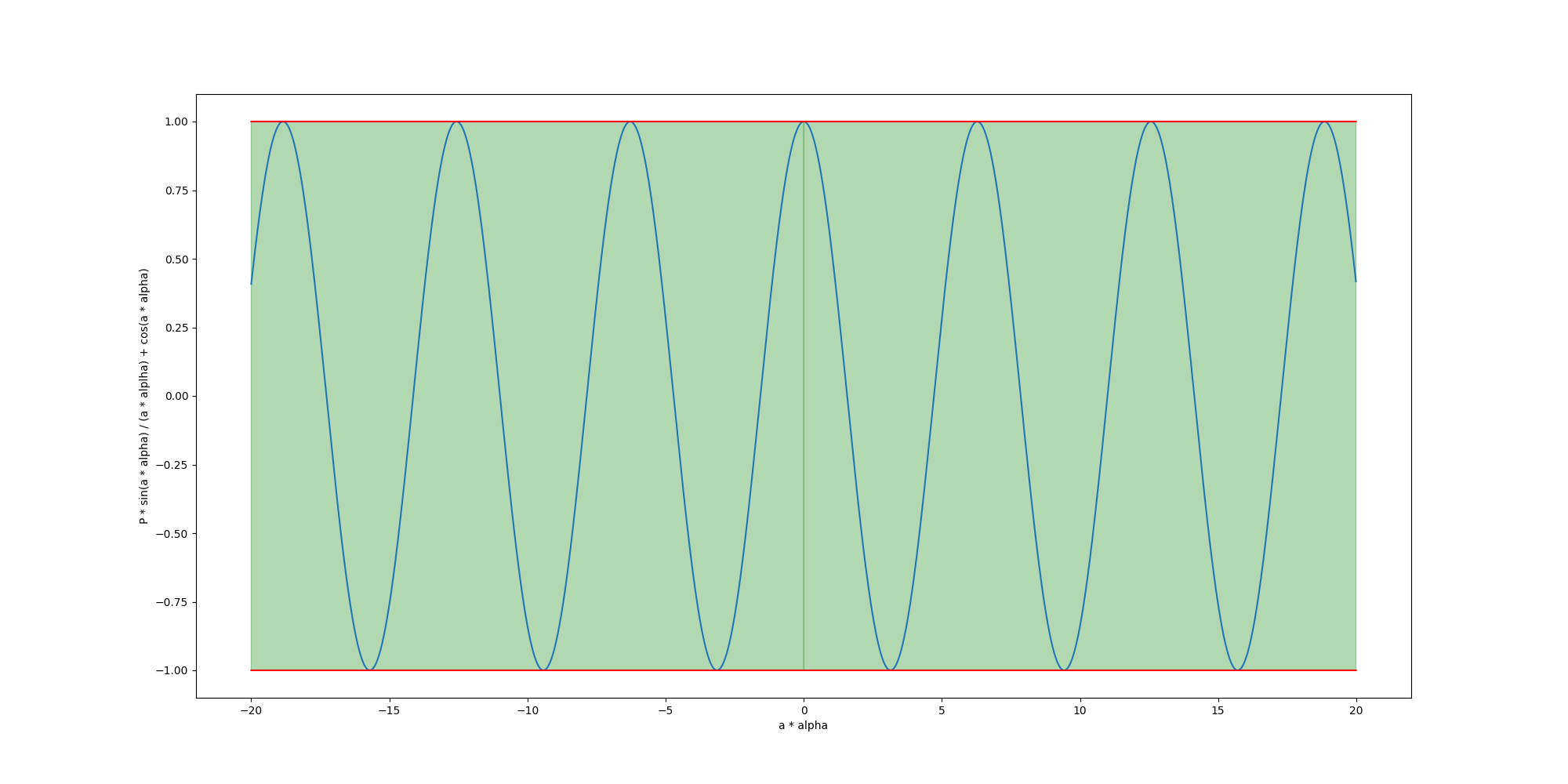
Получим уравнение вида:

Данное уравнение называется уравнением Кронига-Пени. Уравнение выражает зависимость энергии электрона, которая входит в коэффициент α, от волнового числа k для барьеров различной прозрачности P. Поскольку не может быть больше ±1(-1 <cos(ka)<=1), то и левая часть уравнения лежит в пределах. Эти значения определяют области разрешенных энергий электрона – энергетические зоны. Они отдельны друг от друга полосами запрещенных энергий – запрещенные зоны. Ширина зон зависит от параметра прозрачности барьера P.

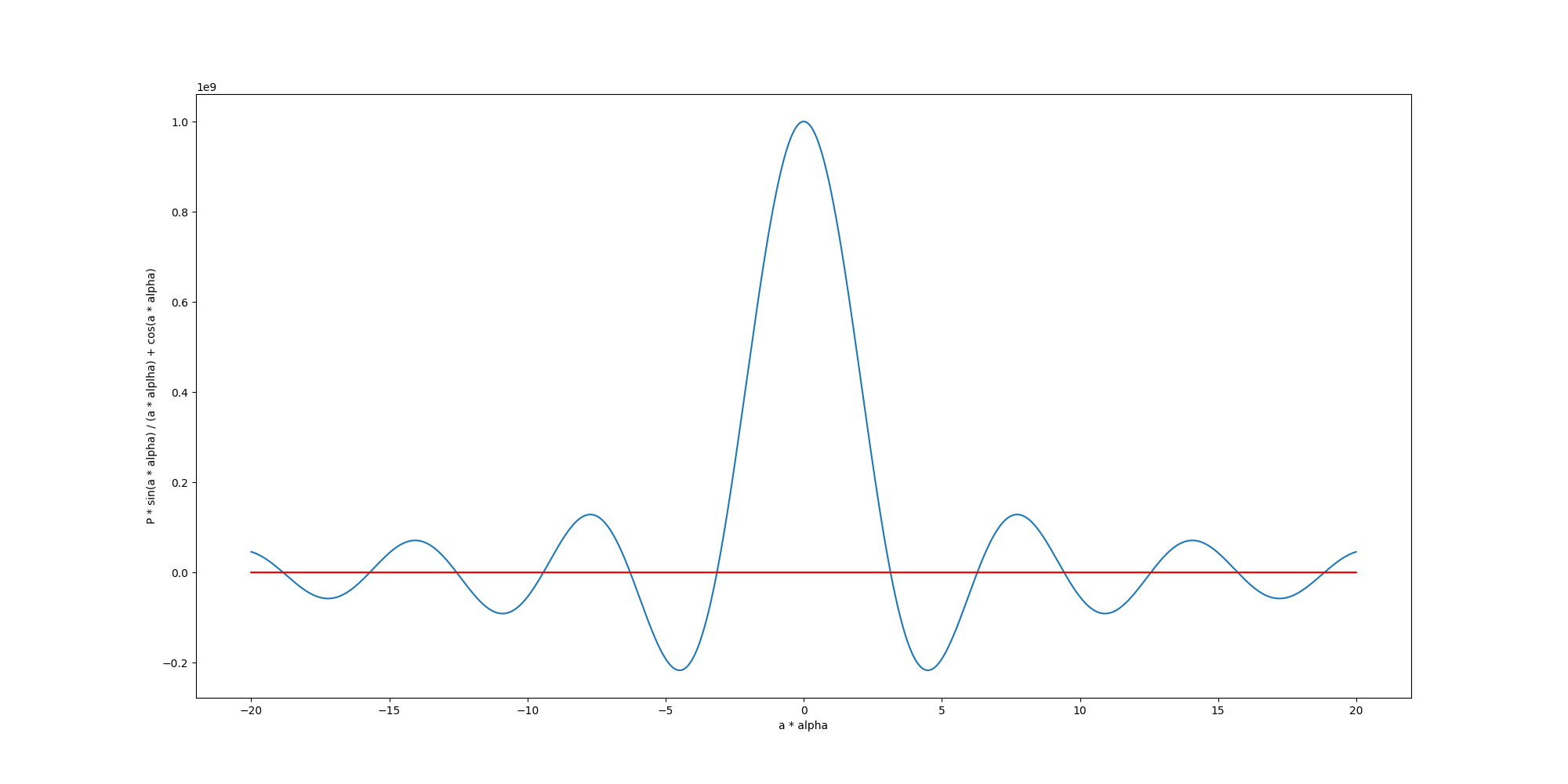
Смоделируем уравнение Кронига-Пени:

import numpy as np  
from math import sin, cos, pi  
import matplotlib.pyplot as plt  
  
P = 10  
L = 40  
step = 0.01  
  
  
def p\_part(x):  
 return P \* sin(x) / x + cos(x)  
  
  
def iterate():  
 start = -L / 2  
 cur = start  
  
 alpha\_a = []  
 f = []  
  
 while cur <= -start:  
 alpha\_a.append(cur)  
 f.append(p\_part(cur))  
 cur += step  
 fmin = [-1 for i in alpha\_a]  
 fmax = [1 for i in alpha\_a]  
  
 return alpha\_a, f, fmin, fmax  
  
  
x, f, minn, maxx = iterate()  
fig = plt.figure()  
fig.set\_figheight(10)  
fig.set\_figwidth(20)  
ax = fig.add\_subplot(111)  
ax.plot(x, f)  
ax.plot(x, minn, color="red")  
ax.plot(x, maxx, color="red")  
  
cond = [1 > f[i] > -1 for i in range(len(f))]  
ax.fill\_between(x, minn, maxx, where=cond, color="green", alpha=0.3)  
  
ax.set\_xlabel("a \* alpha")  
ax.set\_ylabel("P \* sin(a \* alpha) / (a \* alplha) + cos(a \* alpha)")  
plt.show()

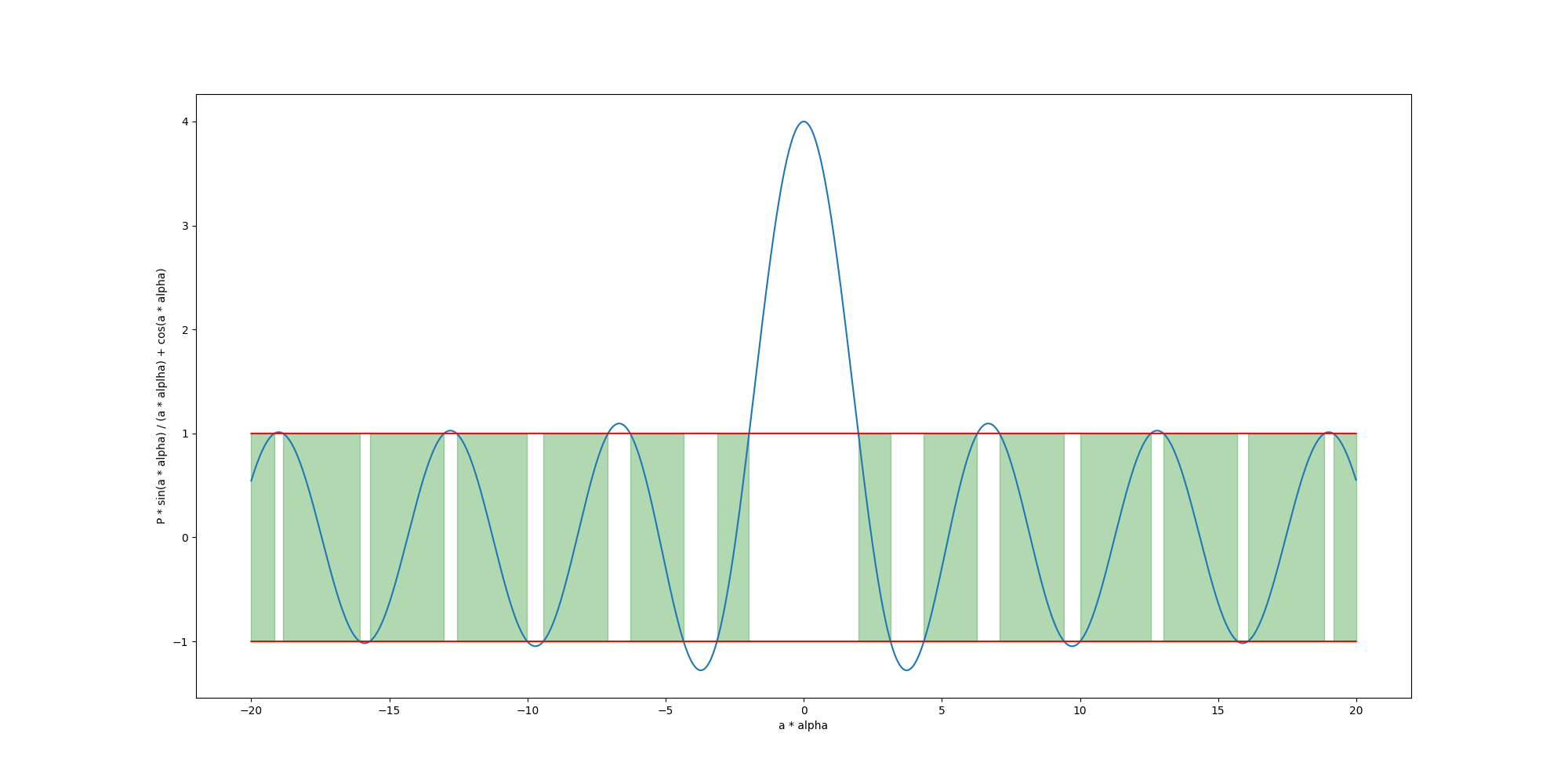
Рассмотрим случай, когда электрон свободен (P=0):



Рассмотрим случай, когда стенки непроницаемы (P→ ꝏ)



Рассмотрим 2 промежуточных случая P=3 и P=10

 Изображение выглядит как текст, Шрифт, диаграмма, дизайн

Автоматически созданное описание

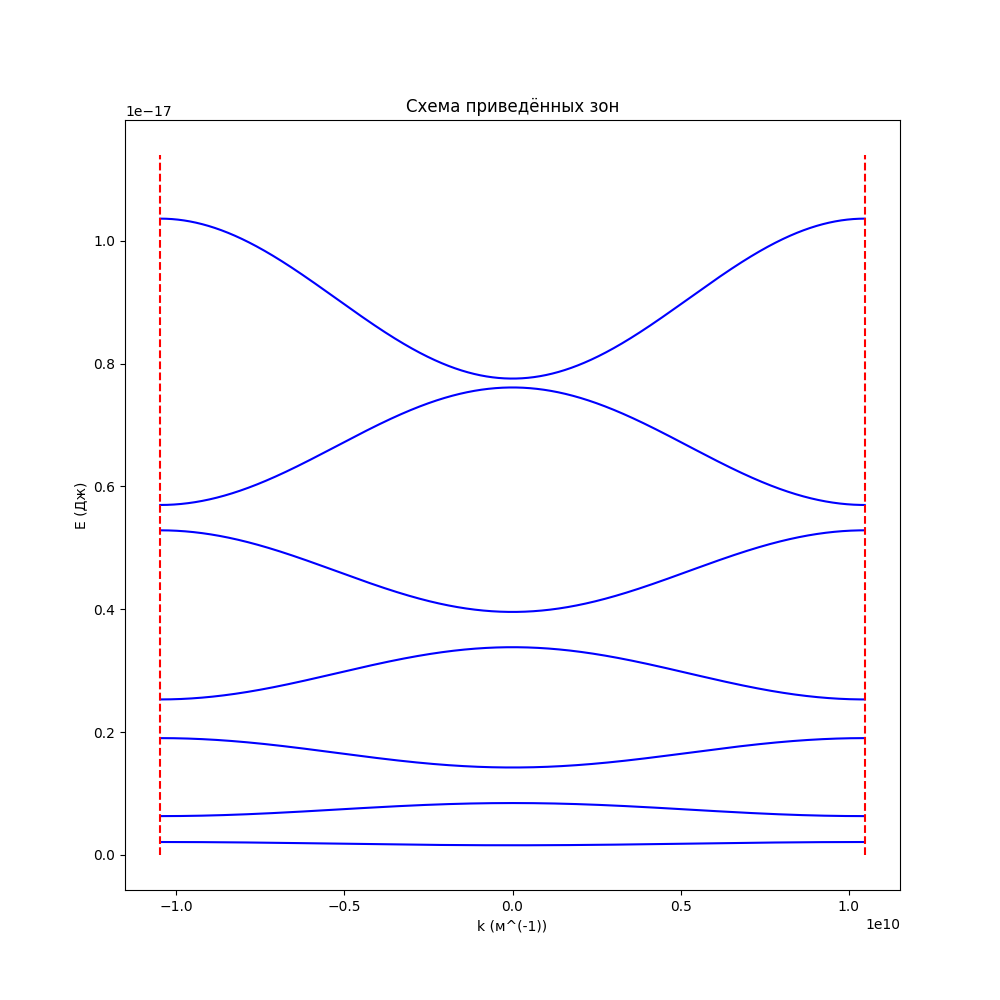
В получившемся уравнении Кронига-Пени мы знаем коэфициент , а значит можно получить зависимость энергии от волнового числа k. Выразить зависимость сложно, но можно предположить, что P>>1, тогда получим, что:

, где – энергия n-ого энергетического уровня электрона в изолированной бесконечно глубокой яме

An – коэффициент определяет амплитуду косинусоидальной зависимости E(k). Предположим, что он равен Cn.

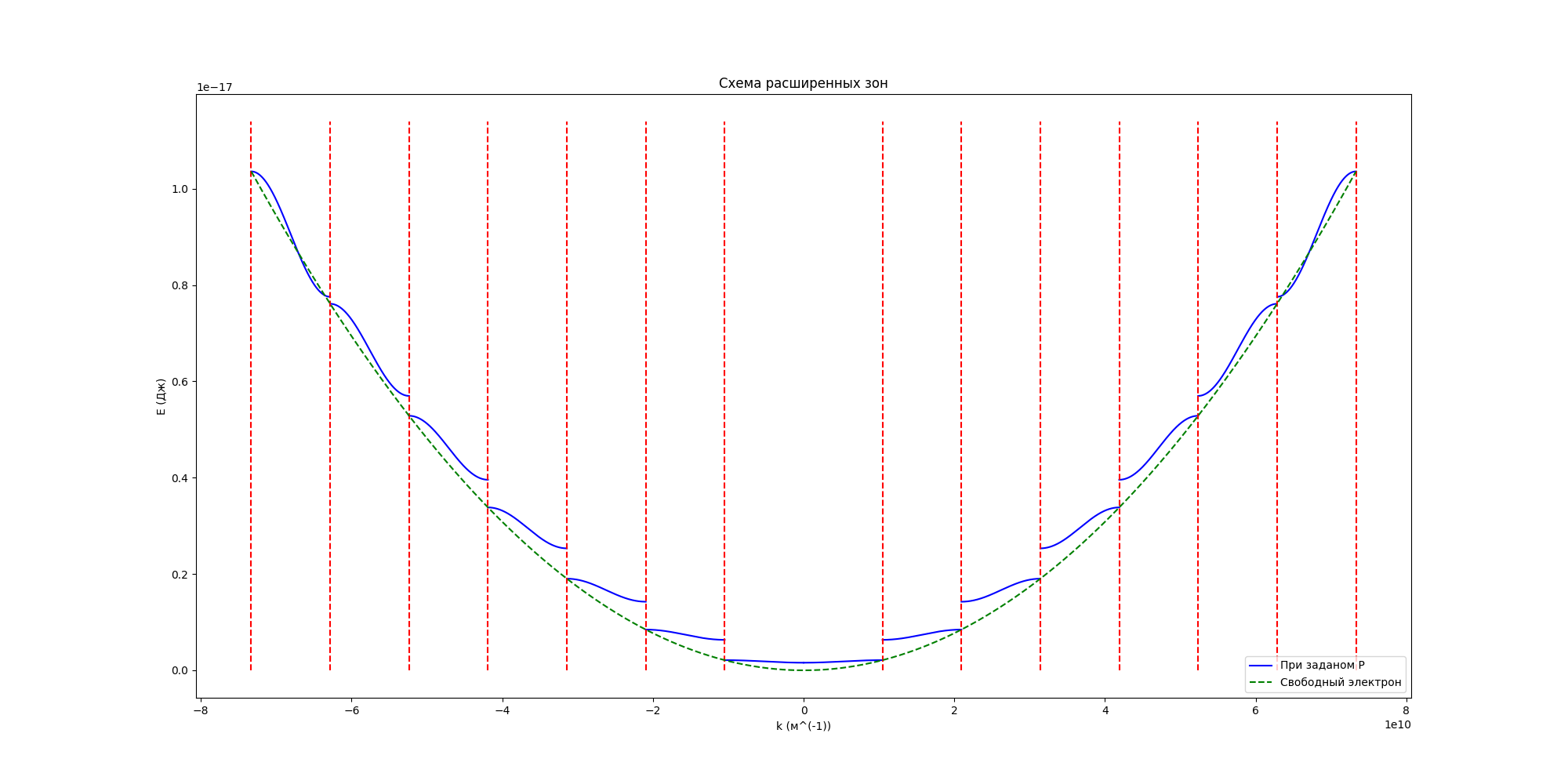
При построении М линий (n=[1,2,…,M]) получаем серию синусоид, с увеличивающейся амплитудой и сменяющейся каждый раз противопожную фазу. Построим M таких синусоидов в промежутке от . Такую модель называют схемой приведенных зон. Она показывает полную информацию о зависимости энергии от волнового числа на первой зоне Бриллюэна.

Рассмотрим случай, когда P=50, a = 3e-10

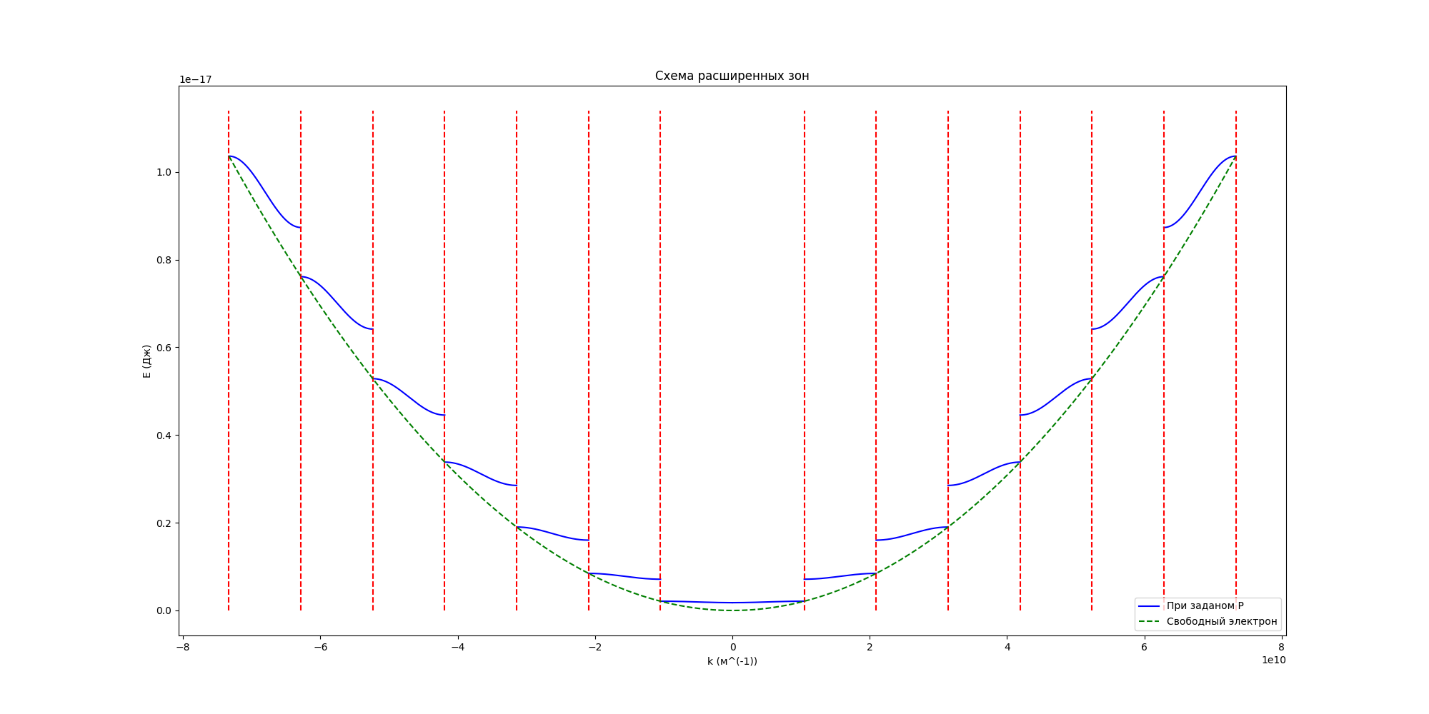
**

Построим М синусоид в промежутке от . Такую модель называют схему расширенных зон.

Возьмем P=50, a=3e-10

**

P=80,a=3e-10



P=1000000,a=3e-10

Изображение выглядит как текст, диаграмма, линия, График

Автоматически созданное описание

1. Выводы:  
   С увеличением энергии электрона ширина разрешенных зон увеличивается, а запрещенных зон уменьшается. При P→ ꝏ разрешенные зоны сужаются, превращаясь в дискретные уровни, соответствующие αα = πn, где n = ±1, ±2, … Тем самым мы приходим к случаю электрона в изолированном атоме. При стремлении прозрачности барьера к нулю P→0, наоборот исчезают зоны, и электрон становится свободным.
2. Код

import numpy as np  
from math import sin, cos, pi  
import matplotlib.pyplot as plt  
  
P = 10  
L = 40  
step = 0.01  
def p\_part(x):  
 return P \* sin(x) / x + cos(x)  
def iterate():  
 start = -L / 2  
 cur = start  
  
 alpha\_a = []  
 f = []  
  
 while cur <= -start:  
 alpha\_a.append(cur)  
 f.append(p\_part(cur))  
 cur += step  
 fmin = [-1 for i in alpha\_a]  
 fmax = [1 for i in alpha\_a]  
  
 return alpha\_a, f, fmin, fmax  
x, f, minn, maxx = iterate()  
fig = plt.figure()  
fig.set\_figheight(10)  
fig.set\_figwidth(20)  
ax = fig.add\_subplot(111)  
ax.plot(x, f)  
ax.plot(x, minn, color="red")  
ax.plot(x, maxx, color="red")  
  
cond = [f[i] < 1 and f[i] > -1 for i in range(len(f))]  
ax.fill\_between(x, minn, maxx, where=cond, color="green", alpha=0.3)  
  
ax.set\_xlabel("a \* alpha")  
ax.set\_ylabel("P \* sin(a \* alpha) / (a \* alplha) + cos(a \* alpha)")  
plt.show()  
  
M = 7  
  
P = 1000000000  
  
h = 1.05e-34  
m = 9.1e-31  
  
a = 3e-10  
n = [i for i in range(1, M + 1)]  
A = [((h \* pi \* i) \*\* 2) / (m \* (a \*\* 2) \* P) for i in n]  
dk = 2 \* M \* pi / a / 100000  
  
  
def energy(k, n):  
 return ((h \*\* 2) \* pi \* (n \*\* 2)) / (2 \* m \* a \*\* 2) - ((h \* pi \* n) \*\* 2) / (m \* (a \*\* 2) \* P) + ((-1) \*\* n) \* A[  
 n - 1] \* cos(k \* a)  
  
  
def iterate\_zones\_mini():  
 start = -pi / a  
 cur = start  
  
 E = []  
 ind = 0  
 k = []  
  
 for i in n:  
 E.append([])  
 while cur <= -start:  
 E[ind].append(energy(cur, i))  
  
 if i == 1:  
 k.append(cur)  
  
 cur += dk  
  
 ind += 1  
 cur = start  
  
 return k, E  
  
  
k, E = iterate\_zones\_mini()  
fig = plt.figure()  
fig.set\_figheight(10)  
fig.set\_figwidth(10)  
ax = fig.add\_subplot(111)  
for i in range(len(n)):  
 ax.plot(k, E[i], color="blue")  
  
maxh = max(E[len(E) - 1]);  
  
ax.vlines(-pi / a, 0, maxh \* 1.1, color="red", ls="--")  
ax.vlines(pi / a, 0, maxh \* 1.1, color="red", ls="--")  
ax.set\_xlabel("k (м^(-1))")  
ax.set\_ylabel("E (Дж)")  
ax.set\_title("Схема приведённых зон")  
plt.show()  
def iterate\_zones\_maxi():  
 start = -max(n) \* pi / a  
 k = []  
 E = []  
 cur = start  
 cur\_n = max(n)  
 ind = 0  
 while cur\_n > 0:  
 k.append([])  
 E.append([])  
 while cur <= (-cur\_n + 1) \* pi / a:  
 k[ind].append(cur)  
 E[ind].append(energy(cur, cur\_n))  
 cur += dk  
 cur\_n -= 1  
 ind += 1  
  
 ind = max(n)  
 cur\_n = 1  
 while cur <= -start:  
 k.append([])  
 E.append([])  
 while cur <= cur\_n \* pi / a:  
 k[ind].append(cur)  
 E[ind].append(energy(cur, cur\_n))  
 cur += dk  
 cur\_n += 1  
 ind += 1  
  
 return k, E  
  
  
def free\_electron(kx, Ex):  
 k = []  
 E = []  
 kn = []  
 En = []  
 over\_zero = False  
  
 for i in range(len(Ex)):  
 maxi = max(Ex[i]);  
 mk = kx[i][Ex[i].index(maxi)]  
 if (mk > 0 and not over\_zero):  
 k.append(0)  
 E.append(0)  
 over\_zero = True  
  
 k.append(mk)  
 E.append(maxi)  
  
 x1, x2, x3 = k[0], k[1], k[2]  
 y1, y2, y3 = E[0], E[1], E[2]  
  
 ap = (y3 - (x3 \* (y2 - y1) + x2 \* y1 - x1 \* y2) / (x2 - x1)) / (x3 \* (x3 - x1 - x2) + x1 \* x2)  
 b = (y2 - y1) / (x2 - x1) - ap \* (x1 + x2)  
 c = (x2 \* y1 - x1 \* y2) / (x2 - x1) + ap \* x1 \* x2  
  
 start = -max(n) \* pi / a  
 cur = start  
  
 while cur <= -start:  
 kn.append(cur)  
 En.append(ap \* cur \*\* 2 + b \* cur + c)  
 cur += dk  
  
 return kn, En  
  
  
kx, Ex = iterate\_zones\_maxi()  
fig = plt.figure()  
fig.set\_figheight(10)  
fig.set\_figwidth(20)  
ax = fig.add\_subplot(111)  
  
ax.plot(kx[0], Ex[0], color="blue", label="При заданом P")  
for i in range(1, len(kx)):  
 ax.plot(kx[i], Ex[i], color="blue")  
  
maxh = max(Ex[len(Ex) - 1]);  
  
for i in n:  
 ax.vlines(-i \* pi / a, 0, maxh \* 1.1, color="red", ls="--")  
 ax.vlines(i \* pi / a, 0, maxh \* 1.1, color="red", ls="--")  
  
fk, fE = free\_electron(kx, Ex)  
ax.plot(fk, fE, color="green", ls="--", label="Свободный электрон")  
  
ax.set\_xlabel("k (м^(-1))")  
ax.set\_ylabel("E (Дж)")  
ax.set\_title("Схема расширенных зон")  
ax.legend(loc="lower right")  
plt.show()