«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра мікроелектроніки

Лабораторна робота №22

Варіант №21

Виконав: студент групи ДП-82

Мнацаканов Антон

Перевірив: Домбругов М.Р.

Київ-2020

**Мета роботи:** вивчення методики розв’язання крайових задач методом скінченних різниць (методом сіток).

**Що зробити:** розрахувати енергетичні рівні стаціонарних станів електрона і відповідні їм хвильові функції, розв’язуючи рівняння Шредингера методом сіток. Скористатися процедурою діагоналізації симетричної матриці методом обертань Якобі. Зобразити графічно кілька розрахованих нижніх енергетичних рівней, а також відповідні їм хвильові функції.

**Хід роботи**

**код на С:**

#include <math.h>

#include <stdio.h>

#include <float.h>

#include <stdlib.h>

#define EPS 1e-8

**double** Q = 3.809911;

**double** U1(**double** x, **double** a, **double** b, **double** U\_0){

**double** L = a;

**if**(x<=0)**return** DBL\_MAX;

**if**(x>=L)**return** DBL\_MAX;

**return** 0;

}

**double** L1(**double** a, **double** b){

**return** a;

}

**double** U2(**double** x, **double** a, **double** b, **double** U\_0){

**double** L = a+b+a;

**if**(x<=0)**return** DBL\_MAX;

**if**(x>=L)**return** DBL\_MAX;

**if**((x>0 && x<a) || (x>(a+b) && x<L))

**return** 0;

**return** U\_0;

}

**double** L2(**double** a, **double** b){

**return** a+b+a;

}

**double** U3(**double** x, **double** a, **double** b, **double** U\_0){

**double** L = a+b+a+b+a;

**if**(x<=0)**return** DBL\_MAX;

**if**(x>=L)**return** DBL\_MAX;

**if**((x>0 && x<a) || (x>(a+b) && x<(a+b+a)) || (x>(a+b+a+b) && x<L))

**return** 0;

**return** U\_0;

}

**double** L3(**double** a, **double** b){

**return** a+b+a+b+a;

}

**double**\*\* create\_mat(**int** n){

**double** \*\*res = calloc(n,**sizeof**(**double**\*));

**for**(**int** i=0;i<n;++i){

res[i] = calloc(n,**sizeof**(**double**));

}

**return** res;

}

**double**\*\* make\_gamiltonian(**int** n, **double**(\*U\_func)(**double**, **double**, **double**, **double**), **double**(\*L\_func)(**double** a, **double** b), **double** a, **double** b, **double** U\_0){

**double** L = L\_func(a,b);

**double** h = L/n;

**double** R = Q/(h\*h);

**double**\*\* ham = create\_mat(n-1);

ham[0][0] = (2.0\*R+U\_func(h, a, b, U\_0));

**for**(**int** i = 1; i<n-1; ++i){

ham[i][i] = (2.0\*R+U\_func((i+1)\*h, a, b, U\_0));

ham[i-1][i] = -R;

ham[i][i-1] = -R;

}

**return** ham;

}

**void** print\_mat(**int** n, **double**\*\* m){

**for**(**int** i=0;i<n;i++){

**for**(**int** j=0;j<n;j++){

printf("%e,", m[i][j]);

}

printf("\n");

}

}

**int** jacobi(**int** n, **double**\*\* d, **double**\*\* q, **double** eps){

**for**(**int** i=0; i<n;++i){

**for** (**int** j = 0; j < n; j++){

**if**(i==j){

q[i][j]=1;

}

**else** {

q[i][j]=0;

}

}

}

**do** {

**double** dlm=d[1][0];

**int** l=1;

**int** m=0;

**for** (**int** i = 1; i < n; i++){

**for** (**int** j = 0; j < (i-1); j++){

**if**(fabs(d[i][j]) > fabs(dlm)){

dlm=d[i][j];

l=i;

m=j;

}

}

}

**if** (fabs(dlm)<eps) **break**;

**double** t=(d[l][l]-d[m][m])/(2.0f\*d[l][m]);

**double** u=t/sqrt(1.0f+t\*t) ;

**double** c=sqrt((1.0f+u)/2.0f);

**double** s=sqrt((1.0f-u)/2.0f);

**for**(**int** i=0; i<n; i++){

**double** qil = q[i][l]\*c+q[i][m]\*s;

**double** qim = -q[i][l]\*s+q[i][m]\*c;

q[i][l]=qil;

q[i][m]=qim;

}

**double** dll=d[l][l]\*c\*c+2\*d[l][m]\*c\*s+d[m][m]\*s\*s;

**double** dmm=d[l][l]\*s\*s-2\*d[l][m]\*c\*s+d[m][m]\*c\*c;

d[l][l]=dll;

d[m][m]=dmm;

d[l][m]=0;

d[m][l]=0;

**for** (**int** i = 0; i < n; i++){

**if**(i!=l && i!=m){

**double** dil= d[i][l]\*c+d[i][m]\*s;

**double** dim=-d[i][l]\*s+d[i][m]\*c;

d[i][l]=dil;

d[l][i]=d[i][l];

d[i][m]=dim;

d[m][i]=d[i][m];

}

}

}**while**(1);

**return** 0;

}

**double**\*\* copy\_mat(**int** n, **double**\*\* m){

**double**\*\* res=create\_mat(n);

**for**(**int** i=0;i<n;++i){

**for**(**int** j=0; j<n; ++j){

res[i][j]=m[i][j];

}

}

**return** res;

}

**double**\*\* norm(**int** n, **double**\*\* m, **double** h){

**for**(**int** i=0;i<n;++i){

**for**(**int** j=0; j<n; ++j){

m[i][j]=m[i][j]/sqrt(h);

}

}

**return** m;

}

**typedef** **struct** shredinger\_type{

**double** W;

**double**\* psi;

} shredinger\_t;

**int** shredinger\_comparator(**const** **void**\* p1, **const** **void**\* p2){

**if**(((shredinger\_t\*)p1)->W == ((shredinger\_t\*)p2)->W){

**return** 0;

}

**if**(((shredinger\_t\*)p1)->W > ((shredinger\_t\*)p2)->W){

**return** 1;

}

**return** -1;

}

**void** solve\_shredinger(**int** n, **double**(\*U\_func)(**double**, **double**, **double**, **double**), **double**(\*L\_func)(**double** a, **double** b), **double** a, **double** b, **double** U\_0){

**double**\*\* ham = make\_gamiltonian(n, U\_func, L\_func, a, b, U\_0);

// print\_mat(n-1, ham);

**double**\*\* d = copy\_mat(n-1,ham);

**double**\*\* q = create\_mat(n-1);

jacobi(n-1, d, q, EPS);

//q = norm(n-1, q, L\_func(a,b)/n);

printf("\n");

// print\_mat(n-1, q);

printf("\n");

// print\_mat(n-1, d);

shredinger\_t\* st = malloc(n-1\***sizeof**(shredinger\_t));

**for**(**int** i=0; i<n-1;++i){

st[i].W = d[i][i];

st[i].psi = malloc((n-1)\***sizeof**(**double**));

**for**(**int** j=0;j<n-1;++j){

st[i].psi[j] = q[j][i];

}

}

qsort(st, n-1, **sizeof**(shredinger\_t), shredinger\_comparator);

**for**(**int** i=0; i<10;++i){

printf("W(%i) - %e\n", i, st[i].W);

}

/\* for(int i=0; i<n-1;++i){ \*/

/\* printf("%e ", (i+1)\*(L\_func(a,b)/n)); \*/

/\* for(int j=0;j<n-1;++j){ \*/

/\* printf("%e ",st[j].W ); \*/

/\* } \*/

/\* for(int j=0;j<n-1;++j){ \*/

/\* printf("%e ",st[j].psi[i]+st[j].W); \*/

/\* } \*/

/\* printf("\n"); \*/

/\* } \*/

}

**int** main(){

// solve\_shredinger(100, U3, L3, 8,1,6);

// solve\_shredinger(30, U1, L1, 8,1,6);

**double** W1 = M\_PI\*M\_PI\*Q/L1(8,1)/L1(8,1);

printf("W(1) - %e\n", W1);

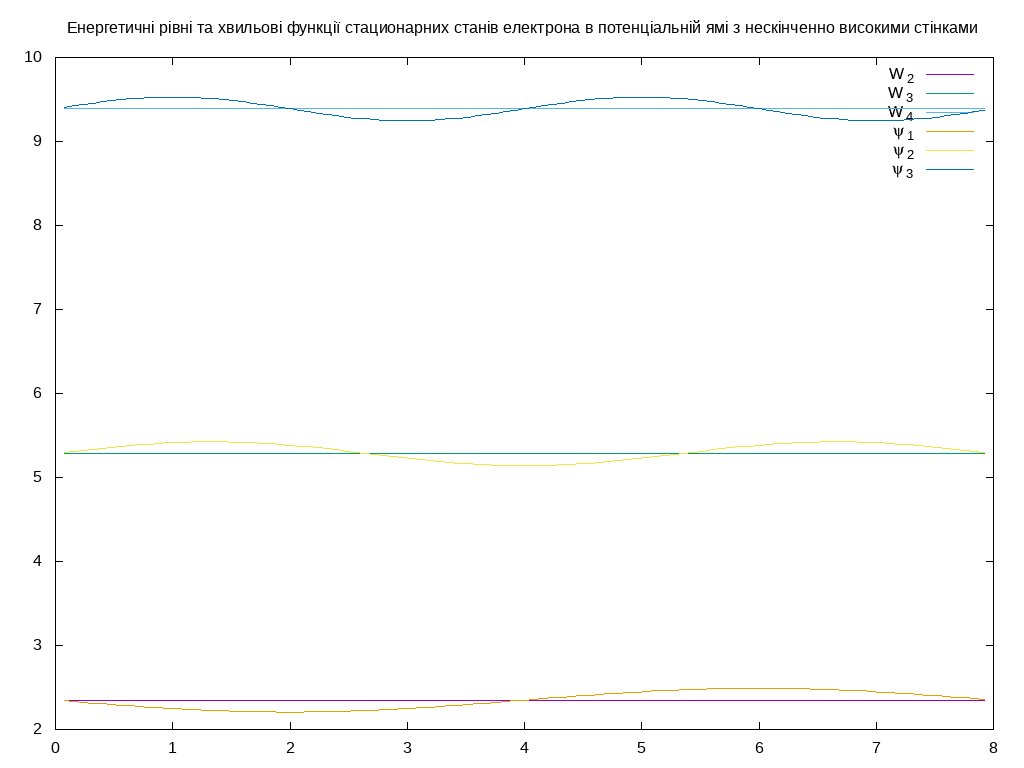
**for**(**int** i=1; i<10;++i){

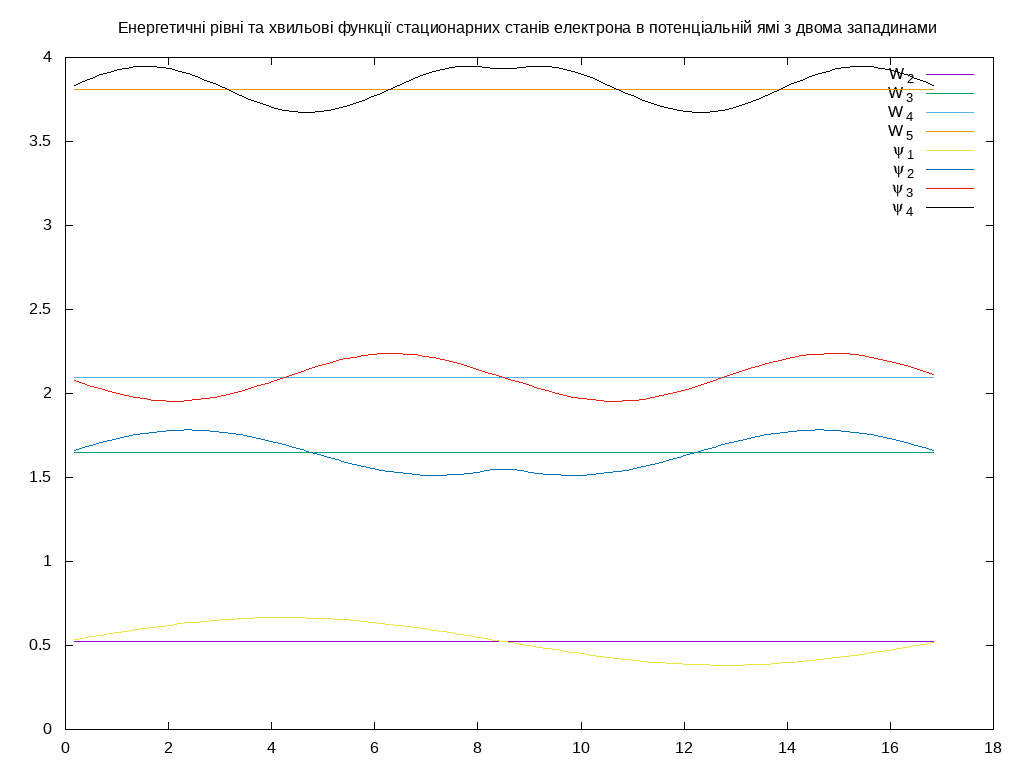
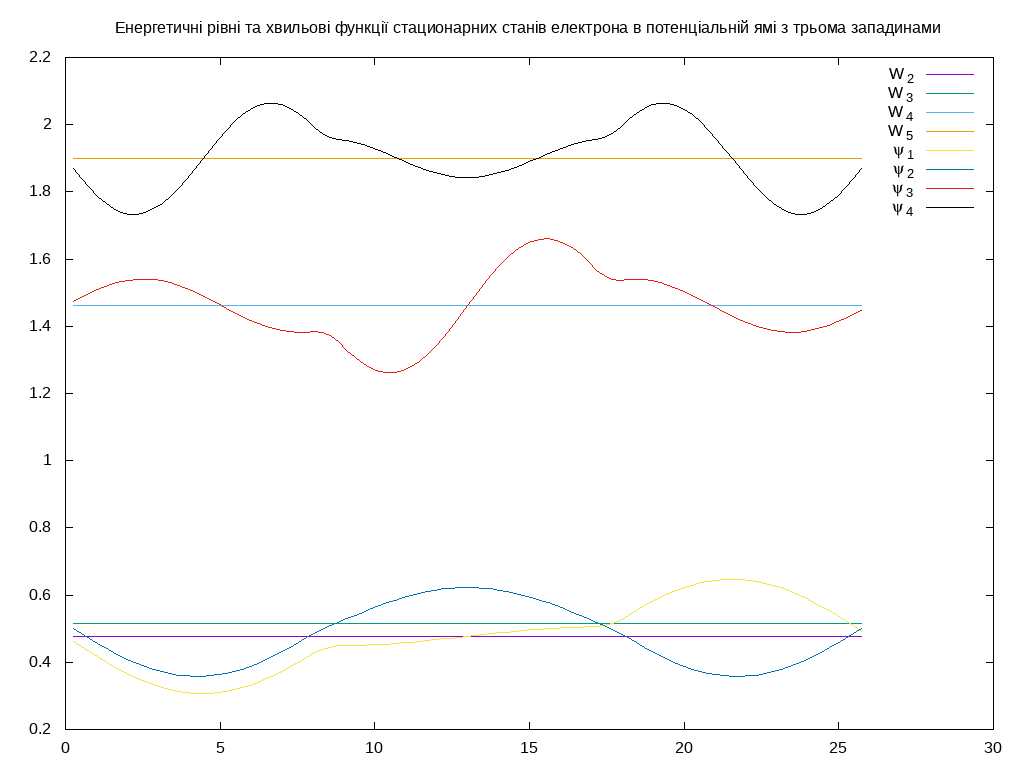
printf("W(%i) - %e\n", i+1, (i+1)\*(i+1)\*W1);

}

**return** 0;

}







**Висновок:** Я написав программу для реализування методики розв’язання крайових задач методом скінченних різниць (методом сіток) на прикладі розв’язання р-ня Шредінгера для поведінки електрона в 1-вимірній потенційній ямі. Якісна поведінка хвильових функцій ψ(k)(x) нижніх енергетичних рівнів за результатами мого дослідження співпали з аналітичною теорією рівнянь типу Шредингера. Токож, додатково – порівняв між собою розраховані значення енергетичних рівней з аналітичними розв’язками ідеалізованої задачі склавши відповідну таблицю.