

Oct 14, 08 10:23

schrodin.cpp

Page 1/4

```

#include <iostream>
#include <cmath>
#include <fstream>
#include <iomanip>
using namespace std;
int sturm(double*, double, int, double);
int tridiag(double *, double *, double *, int, int, double*);
double pot(double);
int wavefun(double *, double, double, double, double, int, int, double*);
//nd-1: numero de intervalos, nd: numero de puntos funcion ondas inc c. c.
// nd-2: numero incognitas; espaciado h
int nd=1001;
double * d =new double[nd];
double e=-1;
double e2=e*e;
double prec=1.e-6;

int main(){
    double x0=-20;
    double x1= 20;
    double h=(x1-x0)/(nd-1);
    double h2=h*h;
    double x=x0;
    int m=6;
    double* v=new double [nd];
    double* lambda= new double[m];
    for(int i=0;i<nd;i++){
        v[i]=pot(x);
        x=x+h;
    }
    // for (int i=0;i<nd;i++) cout<<v[i]<<" ";
    // cout <<endl;
    tridiag(d,v,h2,nd,m,lambda);
    cout<<"valores propios"<<endl;
    for(int i=0;i<m;i++) cout <<i<<" "<<lambda[i]/h2<<endl;
    wavefun(d,e, h,x0, x1, nd,m,lambda);
    return 0;
}

int tridiag (double * d, double * v, double h2, int nd, int m, double* lambda)
{
    //definicion matriz
    cout << "h2="<<h2<<endl;
    for(int i=0;i<nd;i++) d[i]=2.+h2*v[i];

    //cota del intervalo de los valores propios
    double Lambda=max(abs(d[0])+1,abs(d[nd-1])+1);
    for(int i=0;i<nd;i++) {
        double next=abs(d[i]+2);
        if(Lambda<next) Lambda=next;
    }
    double a=-Lambda;
    double b=Lambda;
    cout <<"Lambda="<<Lambda<<endl;
    double x[m+1];
    int fill [m+1];
    for(int i=0;i<m+1;i++) fill[i]=0;
    x[0]=a;
    fill[0]=1;

    //Determinación del intervalo con m valores propios por biseccion
    int nb =sturm(d,e2,nd,b);
    int na= sturm(d,e2,nd,a);
    cout<<"na="<<na<<" nb="<<nb<<" m="<<m<<endl;
    double c;
    int nc;
    // Determinacion del extremo superior conteniendo m valores propios
    while(nb!=m) {
        c=(a+b)/2;

```

Oct 14, 08 10:23

schrodin.cpp

Page 2/4

```

nc=sturm(d,e2,nd,c);
// cout<<c<<" "<<nc<<endl;
if(nc>=m) {
    b=c;
    nb=nc;
}
else if(nc<m) { a=c; na=nc; x[nc-1]=c; fill[nc]=1;}
}

//Extremo superior del intervalo
x[m]=b;
// cout<<x[m]<<endl;
// for (int i=0;i<m;i++) cout<<i<<" "<<x[i]<<fill[i]<<sturm(d,e,nd,x[i])<<endl;

//Calculo de cada uno de los intervalos
for (int i=0;i<m+1;i++){

    a=x[0];b=x[m];
    c=(a+b)/2;
    int nc=sturm(d,e2,nd,c);
    int iter=0;
    while(nc!=i) {
        if (nc>=i) b=c;
        else a=c;
        c=(a+b)/2;
        nc=sturm(d,e2,nd,c);
        // cout<<c<<" "<<nc<<endl;
        iter++;
        if ( iter>200) { cout<<"no convergencia"<<endl; break;}
    }
    x[i]=c;
}

for(int i=0;i<m;i++) cout<<"x["<<i<<"]="<<x[i]<<" nc="<<sturm(d,e2,nd,x[i])<<endl;
// for(int i=0;i<m;i++) cout<<"x["<<i<<"]="<<x[i]<<endl;
//Determinacion de los valores propios en cada intervalo

for(int i=0; i<m;i++) {
    a=x[i]; b=x[i+1];
    int iter=0;
    while(abs(b-a)>prec){
        c=(b+a)/2;
        nc=sturm(d,e2,nd,c);
        // cout<<i<<" "<<nc<<" "<<iter<<endl;
        if(nc==i) a=c;
        else b=c;
        iter++;
        if (iter>50) {cout<<"mas de"<< iter<<" iteraciones en biseccion"<<endl; break;}
    }
    lambda[i]=(b+c)/2;
}
return 0;
}

int sturm(double* d, double e,int n,double x){
    //Calcula el numero de cambios de signo nc de una sucesion de Sturm

    int nc = 0;
    int null=0;
    double q = d[0]-x;
    if (q<0) nc = 1;
    if (q==0) null=1;
    for(int i=1;i<n;i++){
        if (null){

```

Oct 14, 08 10:23

schrodin.cpp

Page 3/4

```

    null=0;
    break;
}
if (q != 0) q=d[i] -x -e/q;
else q=d[i]-x;
if (q==0) null=1;
if (q<0) nc++;
}
return nc;
}

double pot(double x){
    return x*x;
}

int wavefun(double* d,double e, double h,double x0,double x1,
            int nd ,int m ,double* lambda){
    //Impresion en fichero de cada funcion de onda
    double wf[m][nd];
    double h2=h*h;
    //for (int i=0;i<nd;i++) cout<<d[i]<<" "<<endl;
    //cout <<endl;
    for(int i=0;i<m;i++){
        double vp=lambda[i];
        // cout<<"lambda="<<lambda[i]<<" vp= "<<vp<<endl;
        //Radio de matching
        int mat = nd/2;
        //Integracion hacia adelante
        wf[i][0]=0;
        wf[i][1]=1.;
        wf[i][2]=d[2]-vp;
        for(int j=3;j<mat+1;j++){
            wf[i][j]=(d[j-1]-vp)*wf[i][j-1]-wf[i][j-2];
            if(abs(wf[i][j])>1.e15) for(int k=0;k<=j;k++) wf[i][k]=wf[i][k]/1.e15;
        }
        double wi=wf[i][mat];
        //Integracion hacia detras
        wf[i][nd]=0;
        wf[i][nd-1]=1.;
        wf[i][nd-2]=d[nd-2]-vp;
        for(int j=nd-3;j>mat-1;j--){
            wf[i][j]=(d[j+1]-vp)*wf[i][j+1]-wf[i][j+2];
            if(abs(wf[i][j])>1.e15) for(int k=nd-1;k>=j;k--) wf[i][k]=wf[i][k]/1.e15;
        }

        //Renormalizacion de la parte exterior
        double we=wf[i][mat];

        double r=wi/we;
        //cout<<"we="<<we;
        //cout<<" wi="<<wi;
        //cout<<" r="<<r<<endl;
        for(int k=mat;k<nd;k++) wf[i][k]=wf[i][k]*r;

        //Normalizacion funcion de ondas
        double nor=0;
        //N numero subintervalos simpson
        int N = nd / 2;

        for (int j = 0; j < N - 1; j++) {
            nor = nor + 4 * wf[i][2 * j + 1]* wf[i][2 * j + 1] + 2 * wf[i][2 * j + 2]
            ]*wf[i][2 * j + 2];
        }
        nor = nor + 4 * wf[i][2 * N - 1]*wf[i][2 * N - 1];
        nor = nor* h / 3;
        nor=1./sqrt(nor);
        for(int k=1;k<nd;k++) wf[i][k]=wf[i][k]*nor;

```

Oct 14, 08 10:23

schrodin.cpp

Page 4/4

```

}
//Impresion funciones de ondas
double x[nd];
for (int i =0;i<nd;i++) x[i]=x0+i*h;
ofstream fout("wf.txt");
for(int i=0;i<nd;i++){
    fout<<x[i]<<" ";
    for (int j=0;j<m;j++) fout<<wf[j][i]<<" ";
    fout<<endl;
}
fout.close();

return 0;
}

```