Numerical Solutions to Schrodinger's Equation

Mitchell Miller

Physics 240

This assignment explores the properties and applications of Schrodinger's Equation. Several different cases of the 'Particle in a Box' problem were solved using the 'Shooting Method' of computation. The shooting method uses the slope of a function to allow the calculation of the function. The first was a particle in an infinitely deep well of zero potential. This is the simplest case of the particle in a box problem. Next, the properties of a parabolic potential were examined the wave function for this case solved. Thirdly, a general, symmetric potential was solved. Finally, non-symmetric potentials were explored. This case proved the most difficult of the four because the initial conditions necessary for the shooting method were significantly more complicated that those of the previous cases. By exploring each of these cases, the properties and implementation of the shooting method were also thoroughly examined.

1. Introduction

The Schrodinger Equation describes how a particle's quantum state changes at various positions and times. This is a direct result of the quantization of charge, meaning that a particle can only exist at discrete values of energy. Whenever a particle gains or loses energy, it does so by transitioning from its current state to a lower or higher one. In order to represent different physical systems, different potentials are applied to the Schrodinger Equation. There are infinitely many different potentials that can be described, ranging from simple square wells, to parabolic wells, to completely random potential fields. Many of these cases can be solved analytically using common differential-equation solving methods, however some more complicated potentials cannot be solved without the assistance of computational methods. The focus of this assignment is to apply the shooting method to different cases. The shooting method works by calculating a solution to the desired equation with a known initial value and a reasonable guess for the slope[4]. Next, the result is compared to the desired outcome and the slope is adjusted to, hopefully, come slightly closer. These calculations are repeated until the desired results, or at least within practical error, are obtained. This method can be used to solve any differential equation.

2. Theory

For this assignment, the time-independent form of Schrodinger's equation was used which has the following form:

$$\hat{H}\Psi = E\Psi \qquad (1)$$

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial z^2} + V \qquad (2)$$

where $\hat{H}$ is the Hamiltonian Operator and *E* is the energy eigenvalue. Equation (2) represents the Hamiltonian Operator, which is also known as the total energy operator. Operators are a sort of mathematical function which indicates what is meant to be done to the term to its right. In the case of the Hamiltonian, the second derivative of the term is multiplied by a series of constants and added to the potential multiplied by the term. For a zero-potential well of width *a*, Equation (1) can be solved analytically. Solving the second-order differential equation caused by the Hamiltonian, one is left with the following equation[1]:

$$\Psi(z) = A\cos\sqrt{\frac{2EM}{\hbar}}z + B\sin\sqrt{\frac{2EM}{\hbar}}z \qquad (3)$$

Using initial conditions that:

$$\Psi(0) = \Psi(a) = 0 \qquad (4)$$

one can conclude that *A* must be zero and

$$\sqrt{\frac{2EM}{\hbar}}a = n\pi \qquad (5)$$

since a trivial solution is given when *B* is equal to zero. From equation (5), one can derive the form for the energy eigenvalues as[3]:

$$E_n = \frac{n^2h^2}{8a^2M} \qquad (6)$$

In order to calculate *B*, one must make use of the probability distribution function. Since the particle must exist between the two sides of the well, *0* and *a*, the following must be true[1]:

$$1 = \int_0^a (B\sin\frac{n\pi}{l}z)^2 dz \qquad (7)$$

By invoking the double angle formula, this can be greatly simplified and then easily integrated to find

$$B = \sqrt{\frac{2}{a}} \qquad (8)$$

This results in the final form for the solution to the particle in a box:

$$\Psi(z) = \sqrt{\frac{2}{a}}\sin\frac{n\pi}{l}z \qquad (9)$$

In order to solve more complex potential fields, computational methods must be implemented. The shooting method is a simple way to solve the Schrodinger Equation with computer programming. Through simple algebraic approximations of derivatives, the following approximations for the first and second derivative can be found

$$\frac{df}{dz} = \frac{f(z+\delta z) - f(z-\delta z)}{2\delta z} \qquad (10)$$

$$\frac{d^2f}{dz^2} = \frac{f(z+\delta z) - 2f(z) + f(z-2\delta z)}{(\delta z)^2} \qquad (11)$$

By inserting equation (11) into equation (1) and simplifying, one arrives at the shooting equation for the Schrodinger Equation:

$$\Psi(z+\delta z) = \left[\frac{2M}{\hbar^2}(\delta z)^2(V(z)-E) + 2\right]\Psi(z) - \Psi(z-\delta z) \qquad (12)$$

Using this, one can solve the Schrodinger Equation for any kind of potential field and any energy. The result is only correct, however, when the energy is equal to an eigenvalue produced by equation (6). These solutions are distinguished from the others by being zero at infinity. With this knowledge, it is simple to pick out the correct solutions from the incorrect.

3. Experimental Method

The first part of this lab examines an infinitely deep quantum well of zero potential inside and infinite outside. By analytically calculating the solutions for the wave function and the energy eigenvalues in equations (6) and (9), one can simply input the desired values. In this program, the user enters the desired values for both the mass of the particle and the width of the box. The program uses equation (6) and (9) to calculate the eigenvalues and the normalized wave function for the first seven quantum states.

The next part of the assignment explores the difference between even and odd parity (symmetric and anti-symmetric) solutions for a parabolic potential. Using the shooting method and equation (12), the even-parity energy eigenvalues were calculated. To calculate the even-parity values, the initial conditions had to be modified to the following:

$$\Psi(0) = 1$$

$$\Psi(\delta z) = \frac{M}{\hbar^2}(\delta z)^2[V(0) - E] + 1$$

Next, the shooting method was applied to a general symmetric potential field. To do this, the user generates a two-column list of position in angstroms and potential in joules separated by a 'tab' delimiter[2],[5]. Then, the program breaks each line into the position and potential, converts the strings to floating point numbers and stores them in two separate arrays. It then calculates δz by finding the difference between the two first terms and uses the potential specified in the file for *V(*δz*)*. It repeats this for all the values in the array, and outputs the solution.

The final part of this assignment examines a non-symmetric potential[6]. The attached program calculates the solution for a simple linear potential difference using the shooting method again. In this program, the *V(z)* in equation (12) is replaced with a function

$$V(z) = C * z$$

This generates a potential field with constant slope *C*.

4. Data and Analysis

The first part of the assignment successfully calculated eigenvalues and wave functions for an infinitely deep zero potential well. For a well of width 1 Å and an electron with mass 9.109534e-31 kg, the first four energy eigenvalues are:

$$E_1=37.60 \text{ eV}$$
$$E_2=150.4 \text{ eV}$$
$$E_3=338.4 \text{ eV}$$
$$E_4=601.6 \text{ eV}$$

These values clearly increase by the square of the quantum state. This is exactly what is predicted by equation (6), which shows that the energy is a constant based on the conditions of the system multiplied by the square of the quantum state.

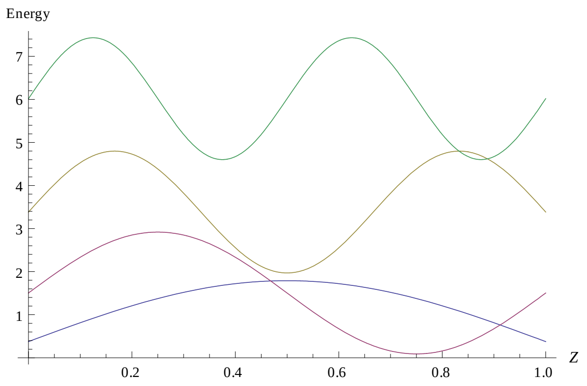


Fig 1. – This figure displays the first Four wave functions for an electron in a 1 Å wide quantum well. Each wave function is offset by its corresponding energy eigenvalues.

For each wave function, a node means that there is no possibility of the particle existing there. This implies that the particle must exist in both sides of the node at all times until it is observed.

Next, the solutions for even-parity eigenvalues were calculated for a parabolic potential. The shooting method was used to calculate the wave function at an effective infinity.

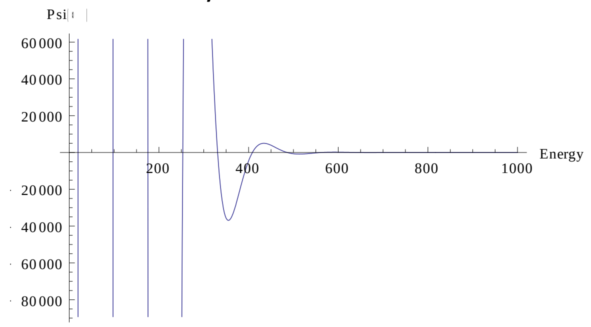


Fig 2. – This figure displays the results for Task 1.5, the energy eigenvalues for a parabolic potential. Whenever the axis is crossed, the zero is a possible eigenvalues for the particle.

With careful observation, it is clear what eigenvalues are allowable. Whenever the function crosses the x-axis, the output of Ψ at infinity is zero. This is the necessary condition for the wave function to have a solution. The first four solutions for a parabolic potential are:

$$E_1 = 19.53 \text{ meV}$$
$$E_2 = 97.52 \text{ meV}$$
$$E_3 = 175.4 \text{ meV}$$
$$E_4 = 253.2 \text{ meV}$$

These values increase proportionally to the square of the quantum state, but not exactly. This is most likely due to the inherent estimation in the computational method. Since each value is approximated, it follows that the solutions would be approximations as well.

5. Conclusion

This lab successfully demonstrated the properties of the wave function and method of implementation of the shooting method. First, the particle in a box problem was solved for an infinitely deep well and a parabolic potential. Next, a user generated general potential was solved, and finally, an asymmetric potential was solved.

6. Bibliography

[1] "Particle in a One Dimensional Box." *McHenryCom Company*. Web. 12 Feb. 2010. <http://user.mc.net/~buckeroo/PODB.html>.

[2] "Reading a File Line By Line - C Code Snippet." *DaniWeb IT Discussion Community*. Web. 12 Feb. 2010. <http://www.daniweb.com/code/snippet216411.html#>.

[3] "Schrodinger equation." *Test Page for Apache Installation*. Web. 12 Feb. 2010. <http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/pbox.html>.

[4] "Shooting Method: Boundary Value Ordinary Differential Equations." *Transforming Numerical Methods Education for the STEM(Science Technology Engineering Mathematics) Undergraduate*. Web. 12 Feb. 2010. <http://numericalmethods.eng.usf.edu/topics/shooting_method.html>.

[5] "Strtok() - Standard C String & Character - C Programming Reference - eLook.org." *ELook.org - Resource and Information Center. Stop looking and start finding!* Web. 12 Feb. 2010. <http://www.elook.org/programming/c/strtok.html>.

[6] Udal, Andres, Reeno Reeder, Enn Velmre, and Paul Harrison. "Comparison of methods for solving the Schrodinger equation for multiquantum well heterostructure applications." *Proceedings of the Estonian Academy of Sciences, Engineering* (2006): 246-61. Print.

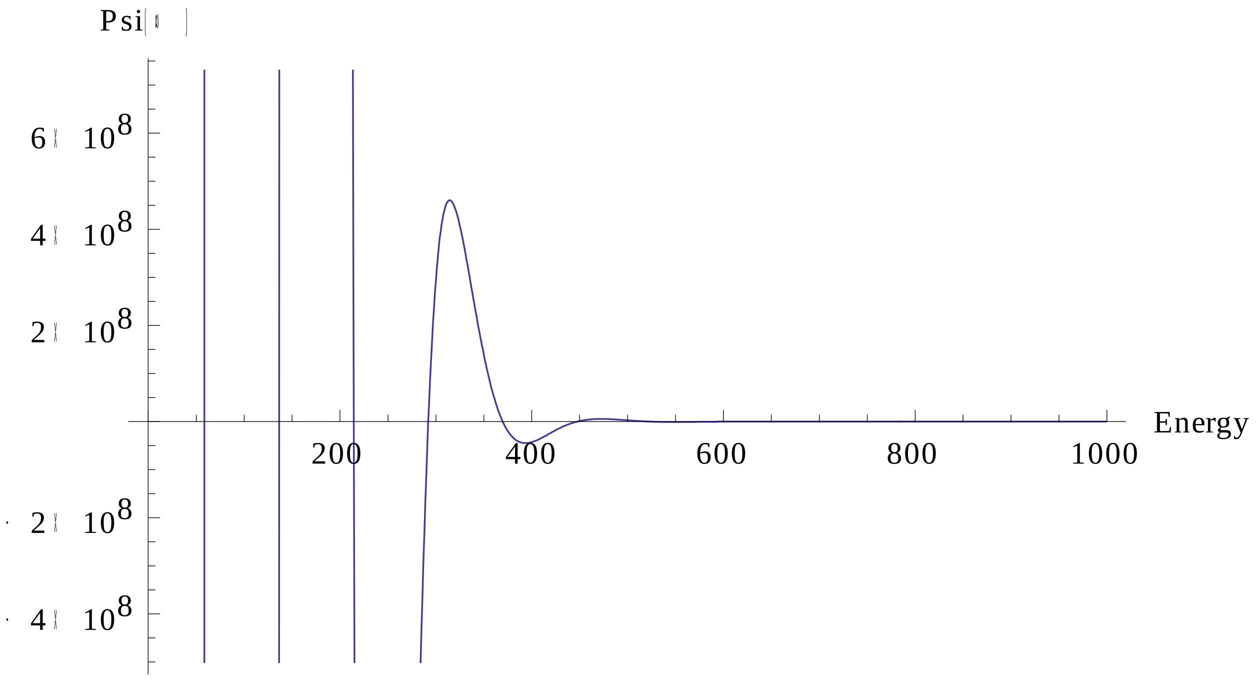
APPENDIX A: TABLES AND FIGURES

Table 1 – Project 1.1 Energy Eigenvalues

Odd-Parity Eigenvalue (meV)	Even-Parity Eigenvalue (meV)
58.541004	58.538734
136.489772	136.485912
214.338727	214.333270
292.083960	292.076545
369.737545	369.728787
447.286694	447.275436
524.734330	524.720499
602.076271	602.060716
679.341655	679.318065
756.622100	756.589709
834.512599	834.461659
914.767035	996.767536

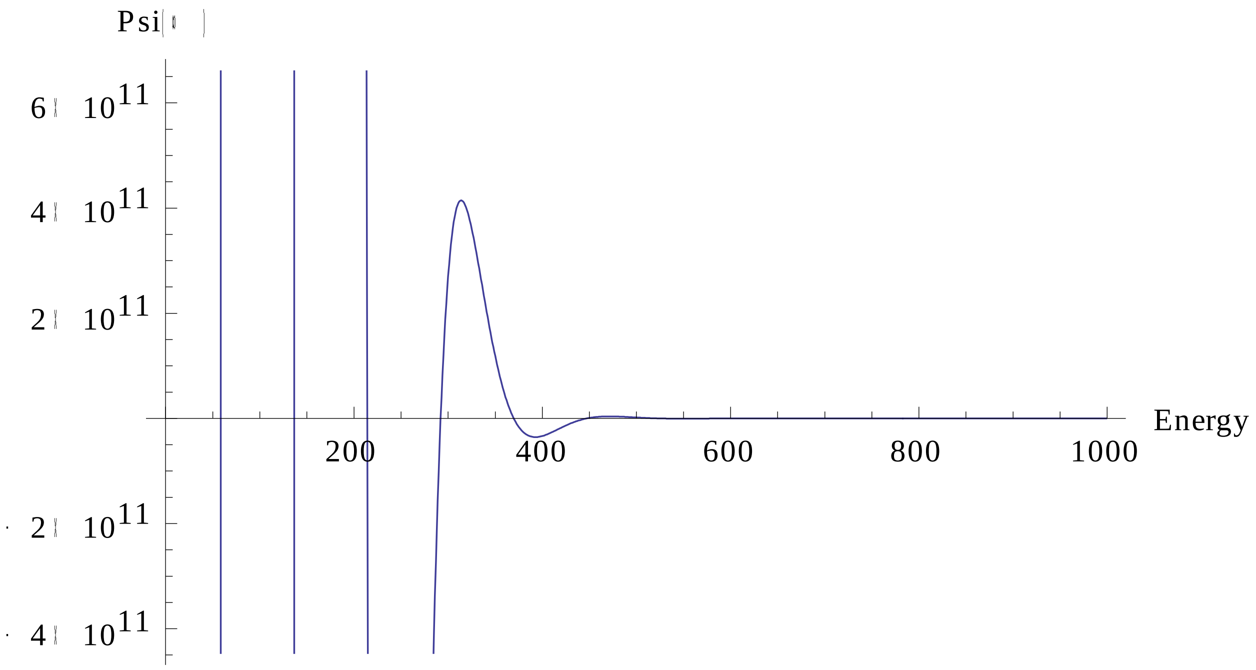
This table shows the energy eigenvalues for both even- and odd-parity solutions for a user generated symmetric potential field. In this case the field is the same as the one used in Task 1.5, a simple parabolic potential.

Figure 3 – Project 1.1 Odd-Parity Psi(Infinity)



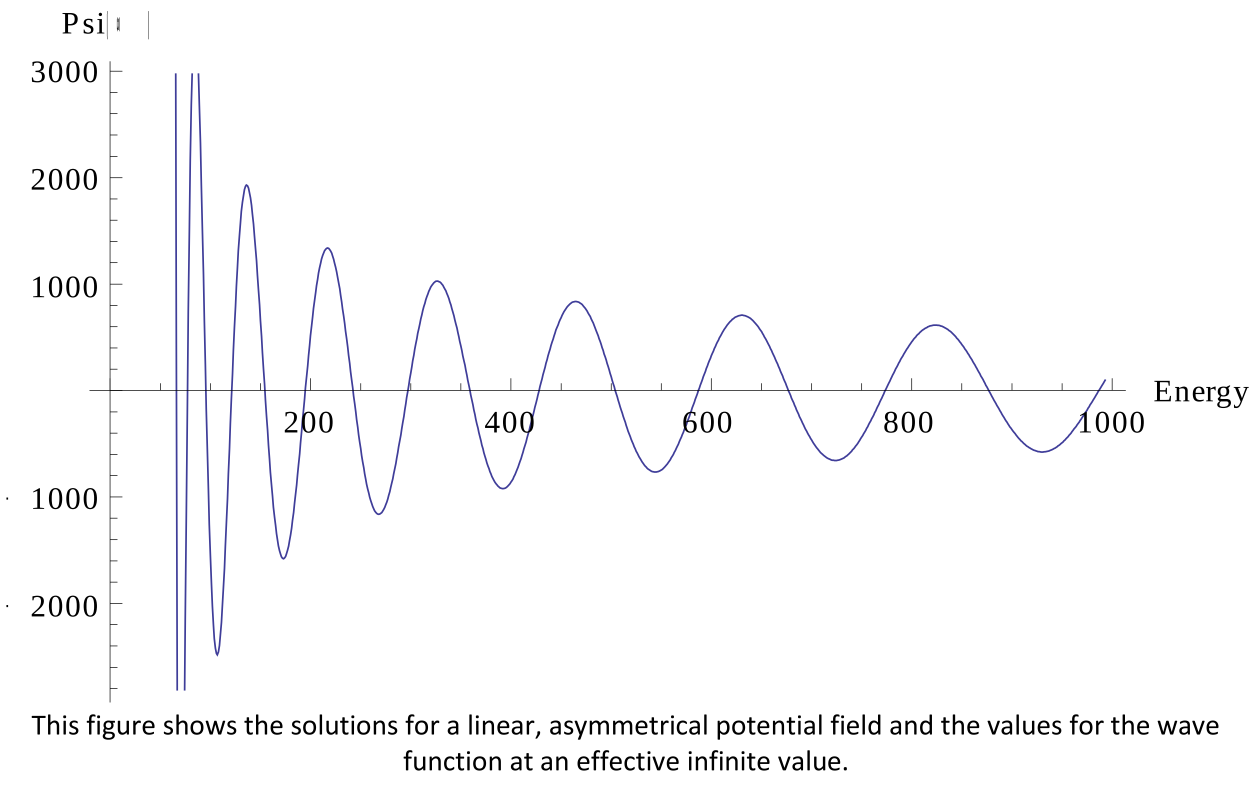
This figure shows the odd-parity values for the wave function at effective infinity for a range of energy.

Figure 4 – Project 1.1 Even-Parity Psi(Infinity)



This figure shows the odd-parity values for the wave function at effective infinity for a range of energy.

Figure 5 – Project 1.2 Non-symmetric Linear Potential Psi(Infinity)



This figure shows the solutions for a linear, asymmetrical potential field and the values for the wave function at an effective infinite value.

APENDIX B: SOURCE CODE

```
#include <stdio.h>
#include <math.h>

//Task 1.1
//Mitchell Miller PHYS240 2/12/10

//**              2     2
//**             n  *  h
//**       E = -------------
//**                    2
//**            8 * m * L
//***********************************************
//**           _________     0.5
//**           |   2          (  n * Pi      )
//**   Psi(x) =  | -----    *   sin(---------- * x)
//**          \|   L          (    L        )


int main(){
        //Initialize variables
        double eigenvalue[7]={0,0,0,0,0,0,0}, planck=6.62606896e-34, hBar=1.054571628e-34, mass=0,
length=0;
   double state=1, internalCoefficient=1, externalCoefficient=1, m=9.10938215e-31, dx=0, x=0;
        int count=1;
        FILE *file;

   file = fopen("Task 11.txt", "w");
        printf("Length = ");
        scanf("%lf",&length);
        printf("Mass = ");
        scanf("%lf",&mass);
        //Calculate eigenvalues
        do{
                dx = 0;
      eigenvalue[count] = (state*state*planck*planck) / (8*mass*length*length);
                printf("Eigenvalue %i = %lf \n", count, eigenvalue[count]*(6.24150974e18));
   //Calculate coefficients and display wave function
                internalCoefficient = state*3.141592653589793/length;
                externalCoefficient = sqrt(2/length);
                while(dx<length){
            x = externalCoefficient*sin(internalCoefficient*dx);
            dx = dx + length/500;
            fprintf(file, "{%e,%e},", dx, x);
      }
      printf("Normalized Wave Function %i = %lf*sin[%lf*x] \n", count, externalCoefficient,
internalCoefficient);
                fprintf(file, "\n\n\n");
```

```
                        state ++;
                        count ++;
            }while(state<7);

            printf("Press Enter to Exit");
            getchar();
            getchar();
            fclose(file);
}




#include <stdio.h>
#include <math.h>

//Task 1.5 (modified version of provided code)
//Mitchell Miller PHYS 240 2/12/10

//**
//**            |                                 |
//**            | 2 * m       2                   |
//**    Psi(dz) = |--------- (dz)  * ( V(0) - E ) + 2 |
//**            |      2                          |
//**            |(h_bar)                          |

//Define constants
#define hbar 1.05459e-34
#define m 9.109534e-31
#define e_0 1.602186e-19

int main(){
   //Intialize variables
   FILE *file;
   FILE *solnfile;
   file = fopen("Task 15.txt", "w");
   solnfile = fopen("Task 15_soln.txt","w");
   float dE=1e-3*e_0;                    //Energy step size
   float dz=1e-10;                       //Position step size
   float E;                              //Energy
   float E_soln,Y1,Y2;
   float psi0,psi1,psi2;                 //Psi_n, Psi_n-1, and Psi_n-2
   float z;                              //Position

   //Begin "shooting method"
   for(E=0; E < e_0; E += dE){
          psi0=1;
          psi1=(m*(dz/hbar)*(dz/hbar)*(e_0*(dz/100e-10)*(dz/100e-10)-E)+1);
```

```
        for(z = dz; z < 100e-10; z += dz){
            //Calculate next value
            psi2 = (2*m*(dz/hbar)*(dz/hbar)*(e_0*(z/100e-10)*(z/100e-10)-E)+2)*psi1-psi0;
            psi0 = psi1;
            psi1 = psi2;
        }
        printf("E=%f meV psi(infinity)=%f \n", E/(1e-3*e_0), psi2);
        fprintf(file, "{%f,%f},", E/(1e-3*e_0), psi2);
        Y2 = psi2;
        if(Y1*Y2<0){
            E_soln = fabs(Y1)*dE/(fabs(Y1)+fabs(Y2))+E-dE;
            fprintf(solnfile,"E = %lf meV\n",E_soln/(1e-3*e_0));
        }
        Y1=Y2;
    }
    printf("Press Enter to Exit");
    getchar();
}



#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

//Project 1.1
//Mitchell Miller PHYS 240 2/12/10

//**          Initial conditions file format:          **\\
//**                                                   **\\
//**  Position in angstroms *TAB* Potential in joules **\\
//**  Position in angstroms *TAB* Potential in joules **\\

//**
//**              |                              |
//**              | 2 * m       2                   |
//**     Psi(dz) = |--------- (dz)  * ( V(dz) - E ) + 2 |
//**              |      2                        |
//**              |(h_bar)                          |

int main(){
        //Initialize variables
        double zInitial[100];                 //Change '100' to number of initial conditions
        double V_zInitial[100];               //Change '100' to number of initial conditions
        int i=0;                              //Counter
        //int n=0;
        double psi0=0,psi1=1,psi2=0;          //Values for Psi_n, Psi_n-1, and Psi_n-2
        double z=0,V_z=0,dz=0;
        double e_0=1.602186e-19;              //Charge of electron
        double dE=1e-3*e_0;                  //Energy step size
        //float dummy=1;
        double E,E_soln;
        double hbar=1.05459e-34;
        double m=9.109534e-31;                //Mass of electron
        memset(zInitial,'\0',100);            //Change '100' to number of initial conditions
        memset(V_zInitial,'\0',100);          //Change '100' to number of initial conditions
        double Y1=1,Y2=1;
FILE *file;
        FILE *outfile;
        FILE *solnfile;
        outfile = fopen("Project 11.txt","w");
        solnfile = fopen("Project 11_soln.txt","w");

        //Begin reading file
        file = fopen("InitialConditions_11.txt", "r");   //Specify path for initial conditions file
        char line[200], *end;
        while(fgets(line, sizeof line, file) != NULL){
                //Store initial condistions from file to array
    zInitial[i] = atof(strtok(line, "\t"));
                V_zInitial[i] = atof(strtok(NULL, "\t"));
                printf("z_i = %f, V(z)_i = %f\n", zInitial[i], V_zInitial[i]);
                i++;
        }

        //Calculate odd parity solutions
        for(E=0; E<e_0;E=E+dE){
    psi1=1;
    psi0=0;
    for(i=0;i<100;i++){
       z = zInitial[i]*1e-10;
       V_z = V_zInitial[i];
       dz = (zInitial[i+1]-zInitial[i])*1e-10;
       psi2 = (2*m*(dz/hbar)*(dz/hbar)*(e_0*V_z-E)+2)*psi1-psi0;      //Calculate next term
       psi0 = psi1;
       psi1 = psi2;
    }
printf("E=%fmeV psi(inf)=%f\n", E/(1e-3*e_0),psi2);
Y2 = psi2;
if(Y1*Y2<0){
    E_soln = fabs(Y1)*dE/(fabs(Y1)+fabs(Y2))+E-dE;
    fprintf(solnfile,"E_odd = %lf meV\n",E_soln/(1e-3*e_0));
    }
Y1=Y2;
fprintf(outfile, "{%f,%e},",E/(1e-3*e_0),psi2);
}
```

```
            fprintf(outfile,"\n\n\n");

            //Calculate even parity solutions
            V_z = V_zInitial[0];
            dz = (zInitial[1]-zInitial[0])*1e-10;
            Y1=1;
            for(E=0; E<e_0;E=E+dE){
        psi1=(m*(dz/hbar)*(dz/hbar)*(e_0*V_z-E)+1);
        psi0=1;
        for(i=0;i<100;i++){
           z = zInitial[i]*1e-10;
           V_z = V_zInitial[i];
           dz = (zInitial[i+1]-zInitial[i])*1e-10;
           psi2 = (2*m*(dz/hbar)*(dz/hbar)*(e_0*V_z-E)+2)*psi1-psi0;      //Calculate next term
           psi0 = psi1;
           psi1 = psi2;
        }
    printf("E=%fmeV psi(inf)=%f\n", E/(1e-3*e_0),psi2);
    Y2 = psi2;
    if(Y1*Y2<0){
        E_soln = fabs(Y1)*dE/(fabs(Y1)+fabs(Y2))+E-dE;
        fprintf(solnfile,"E_even = %lf meV\n",E_soln/(1e-3*e_0));
        }
    Y1=Y2;
    fprintf(outfile, "{%f,%e},",E/(1e-3*e_0), psi2);
    }


            printf("Press Enter to Exit");
            getchar();
            fclose(file);

}



#include <stdio.h>
#include <math.h>

//Project 1.2
//Mitchell Miller PHYS 240 2/12/10

//**
//**            |                              |
//**            | 2 * m       2                  |
//**     Psi(dz) = |--------- (dz)  * ( V(0) - E ) + 2 |
//**            |      2                        |
//**          |(h_bar)                   |

//Define constants
#define hbar 1.05459e-34
#define m 9.109534e-31
#define e_0 1.602186e-19

int main(){
   //Intialize variables
   FILE *file;
   file = fopen("Project 12.txt", "w");
   float dE=1e-3*e_0;                        //Energy step size
   float dz=1e-10;                         //Postion step size
   float E;                              //Energy
   float psi0,psi1,psi2;                     //Psi_n, Psi_n-1, and Psi_n-2
   float z;                              //Position

   //Begin "shooting method"
   for(E=0; E < e_0; E += dE){
         psi0=1e-10;
         psi1=1e-10*pow(2.71828183,dz*pow(2*m*(dz*5e-6-E)/(hbar*hbar),0.5));
         for(z = dz; z < 100e-10; z += dz){
              //Calculate next value
              psi2 = (2*m*(dz/hbar)*(dz/hbar)*(dz*1e-10-E)+2)*psi1-psi0;
              psi0 = psi1;
              psi1 = psi2;
         }
         printf("E=%f meV psi(infinity)=%f \n", E/(1e-3*e_0), psi2);
         fprintf(file, "{%f,%f},", E/(1e-3*e_0), psi2);
   }
   printf("Press Enter to Exit");
   getchar();
}
```