Physics 550 Final Project:

Shooting Quantum Fish

in a Barrel

[I will tell you my name in person when I hand it in]

(***Don't show this to anyone!***)

Abstract:

A shooting algorithm was employed to solve the one-dimensional time-independent Schrödinger wave equation (TISWE) for the infinite square well potential (particle in a box) and the harmonic oscillator. What you won't see here are the fruitless results of 10 hours of labor trying to crack the 1/x potential ☹.

Introduction:

Quantum mechanical systems are governed by the time dependent Schrödinger equation, but if we seek to know a stationary state solution, we can simply examine the TISWE.[1] It is important to solve these equations so that we can understand the *inner workings* of the universe. Of course, the universe we live in is has at least three dimensions. Let's pretend for a while that there is only one.

The infinite square well potential is very useful because many quantum systems can be described as states trapped in a finite region with rigid boundaries. This would include light trapped in an optical cavity.

The harmonic oscillator potential is even more useful. It can be used as the first approximation of any polynomial potential that does not contain a linear term. This would be the case for atomic bonding.

Schrödinger Wave Equation:

The TISWE is given by

$$-\frac{h^2}{8m\pi^2}\left(\nabla^2\psi(x)\right)+V(x)\cdot\psi(x)=E\cdot\psi(x),$$

where in calculations, h=m=π=1. To solve this equation numerically, the following approximation is made:

$$\psi_{n+1}(x)=2\left(V_n(x)-E\right)\cdot\psi_n(x)\cdot\left(\Delta x\right)^2+2\psi_n(x)-\psi_{n-1}(x)^2,$$

where initial conditions will obviously determine the trajectory of the ensuing integration. Additionally, the resultant solution must be square integrable adding a complex boundary condition to the system.

Particle in a Box:

The infinite square well potential is given by

$$V=\infty,\ |x|\geq 1 \text{ and}$$
$$=0,\ |x|<1.$$

Eigenfunction solutions must be symmetric in this potential and must vanish at +/-1 endpoints (and therefore by default be square integrable). Therefore, when an integration is performed, it cannot be known *a priori* if it will satisfy the boundary condition at the endpoints.

Solutions will only exist for certain values of E (the eigenvalues). Therefore E becomes an adjustable parameter wherein integration is computed for a trial E and if a

solution is found, this E is an eigenvalue. If not, a new trial must be performed with another value for E.

A program was created to slowly increase the value of E while an integration trial was performed to see if a satisfactory solution was found (see appendix). Once the first 5 eigenvalues were discovered, the wavefunction solutions, ψ, (the eigenfunctions) were graphed (*Figure 1*).

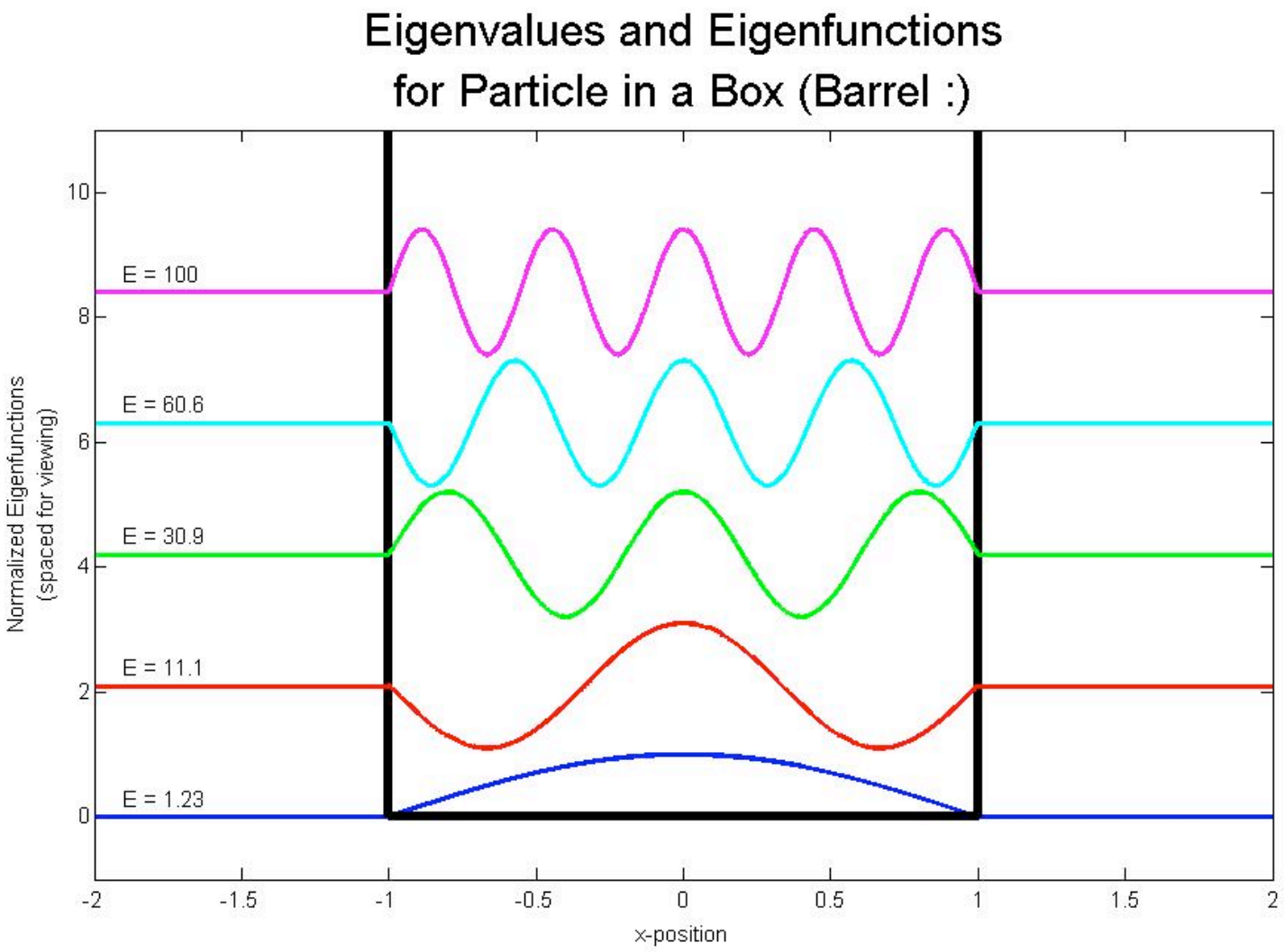

Figure 1: Solutions of Infinite Square Well Potential.

These solutions show the correct form and also based on the number of nodes in each that the program did not skip any possible eigenfunctions. They also match the well-known analytical solutions,

$$\sqrt{2}\sin(n\pi x),\ n = 1,2,3,...^{3}$$

Simple Harmonic Oscillator:

The simple harmonic oscillator potential is given by

$$V = \frac{1}{2}kx^2,$$

with k=25 for this simulation. The boundary value now is more complex. Rather than the wave function simply vanishing at a specific location, now the wavefunction must continually diminish to zero at infinity. Luckily, a small deviation from the true energy eigenstate will quickly result in the integration for the wavefunction becoming unbounded.

Obtaining eigenfunction solutions that achieved the boundary conditions was accomplished by a Newton's method of integration. If at a distant endpoint the solution blew up to positive infinity for one trial energy, but blew up towards negative infinity at another trial energy, the energy interval between them was investigated more finely. This method ran very quickly and could be used to achieve a

high accuracy in the solution for the eigenfunction (*Figure 2*).

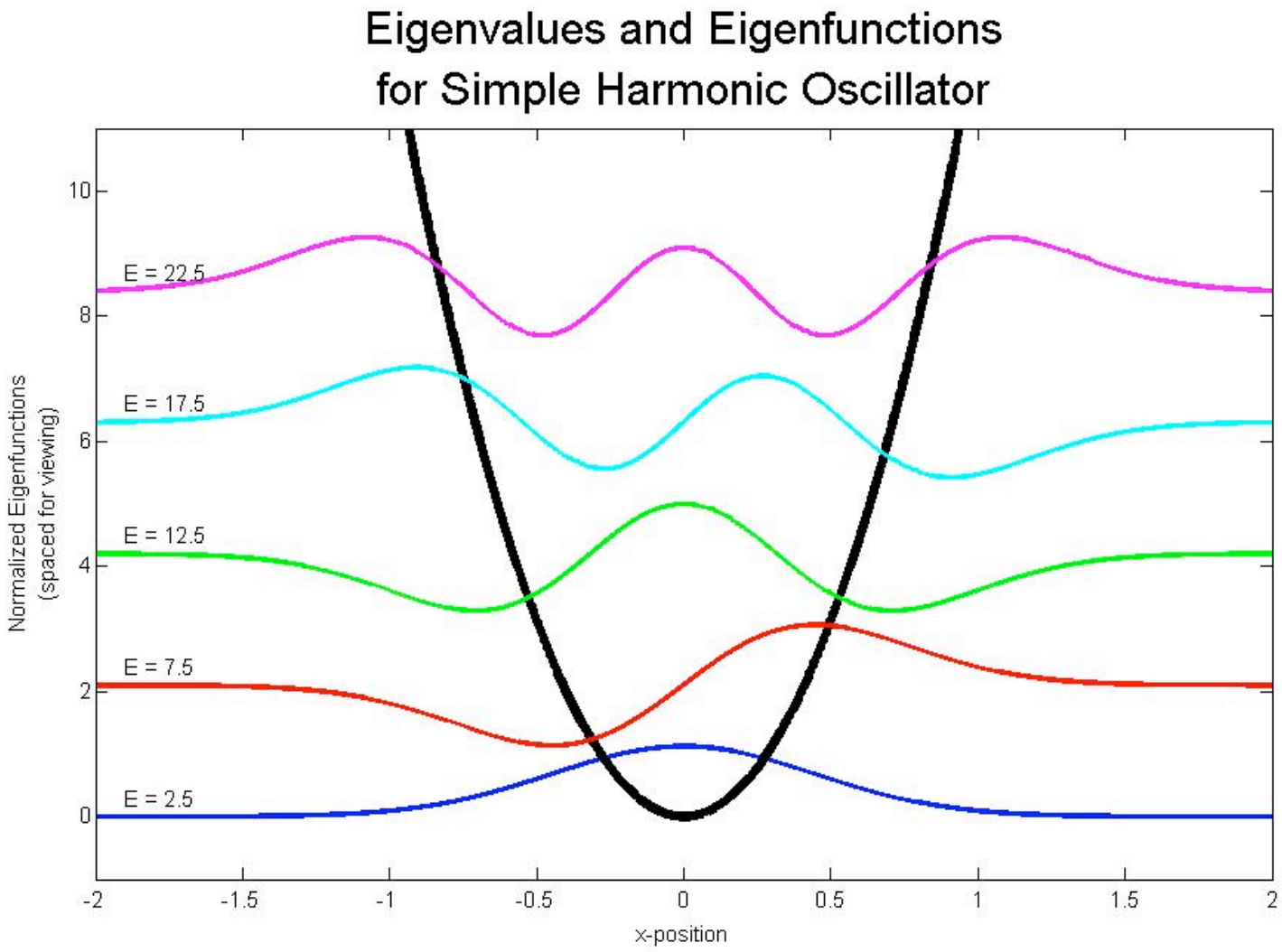

Figure 2: Solutions of Simple Harmonic Oscillator.

The energy eigenstate solutions for the simple harmonic oscillator are well know:

$$E_n = \left(n + \frac{1}{2}\right)\omega = \left(n + \frac{1}{2}\right)\sqrt{\frac{k}{m}} = \left(n + \frac{1}{2}\right)\cdot 5,\ n = 0,1,2,\ldots^{4},$$

and the solutions are in fairly good agreement. Further, no eigenfuction solutions have been skipped and are fairly accurate compared to the analytical solutions.[5,6]

Summary:

Solving the TISWE by shooting methods is difficult because a full integration must be performed before knowing if the boundary conditions have been achieved. One must take care not to miss any eigenstates as the search is conducted. Compared to classical systems, it is not like shooting fish in a barrel.

References:

[1] D. J. Griffiths, *Introduction to Quantum Mechanics, 2nd Ed.,* (Pearson Prentice Hall, Upper Saddle River, NJ, 2005), p. 2.

[2] N. J. Giordano and H. Nakanishi, *Computational Physics, 2nd Ed.,* (Pearson Prentice Hall, Upper Saddle River, NJ, 2006), p. 308.

[3] R. A. Serway, C. J. Moses and C. A. Moyer, *Modern Physics, 3rd Ed.,* (Brooks/Cole-Thomson Learning, Belmont, CA), p. 204.

[4] D. J. Griffiths, *Introduction to Quantum Mechanics, 2nd Ed.,* (Pearson Prentice Hall, Upper Saddle River, NJ, 2005), p. 54.

[5] D. J. Griffiths, *Introduction to Quantum Mechanics, 2nd Ed.,* (Pearson Prentice Hall, Upper Saddle River, NJ, 2005), p. 58.

[6] R. A. Serway, C. J. Moses and C. A. Moyer, *Modern Physics, 3rd Ed.,* (Brooks/Cole-Thomson Learning, Belmont, CA), p. 216.

Borrowed Art:

wacky fish: www.thetalentshow.org

happy box: www.resource.nsw.gov.au

slinky: www.ls-dyna.cn

this and many more scientific portraits at
http://homepage.mac.com/uchii/PhotoAlbum10.html

Appendix:

Program 1: P.I.B.

```
%% Shooting Quantum Fish in a P.I.B. (Barrel).
close all; clear all; prec_end = 2;

set(gcf,'units','normalized','position',[.1,.1,.8,.8]);
set(0,'defaultaxesfontsize',15);
set(0,'defaulttextfontsize',15);
set(0,'defaultlinelinewidth',3);

x = linspace(-2,2,10001); % 21 divisions minimum!
Nx = length(x);
dx = (x(end)-x(1))/(Nx-1);
mid = (Nx-1)/2 + 1;
left = mid - round(1/dx);
right = mid + round(1/dx);
V = x; V(:) = 1e50; V(left+1:right-1) = 0; %%  P.I.B.
P = x;
E = 0;
Ne = 0;

%% Begin search for energy eigenvalues and eigenstates:
while 1,
    E = E + 1;
    flag = 0;
    count = 1;
    while 1,
        E = E + .001;
        P(:) = 0;
        P(mid-1:mid+1) = 1;
        P(right-2:right-1) = 1;
        for i_P = mid+2:right-3,
            P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))*...
               P(i_P-1)-P(i_P-2);
            P(2*mid-i_P) = P(i_P);
        end;
        for i_P = right-2:right-1,
            P(i_P) = round((10^prec_end)*...
                (2*(1-(dx)^2*(E-V(i_P-1)))*P(i_P-1)-...
                P(i_P-2)))/(10^prec_end);
            P(2*mid-i_P) = P(i_P);
        end;
        if (P(right-2:right-1)==[0,0]),
            break;
        end;
        if count==10000,
```

```
        disp('TIMED OUT: eigenvalue not reached!');
        flag = 1;
        break;
    end;
    count = count + 1;
end;
if flag == 1,
    continue;
end;
for i_P = right:Nx,
    P(i_P) = round((10^prec_end)*...
        (2*(1-(dx)^2*(E-V(i_P-1)))*P(i_P-1)-P(i_P-...
           2)))/(10^prec_end);
    P(2*mid-i_P) = P(i_P);
end;
N = sqrt(sum(P.*P)*dx);
P = P/N;
if Ne == 0,
    plot(x,P,'b-');
    axis([-2,2,-1,11]);
    text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
    title({'Eigenvalues and Eigenfunctions',...
        'for Particle in a Box (Barrel :)'},...
        'fontsize',30);
    xlabel('x-position');
    ylabel({'Normalized Eigenfunctions',...
        '(spaced for viewing)'});
    hold on;
    plot([-1,-1,1,1],[100,0,0,100],'k','linewidth',6);
    drawnow;
    Ne = Ne + 1;
elseif Ne == 1,
    P = P + Ne*2.1;
    plot(x,P,'r-');
    text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
    drawnow;
    Ne = Ne + 1;
elseif Ne == 2,
    P = P + Ne*2.1;
    plot(x,P,'g-');
    text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
    drawnow;
    Ne = Ne + 1;
elseif Ne == 3,
    P = P + Ne*2.1;
    plot(x,P,'c-');
    text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
    drawnow;
```

```
        Ne = Ne + 1;
    elseif Ne == 4,
        P = P + Ne*2.1;
        plot(x,P,'m-');
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        drawnow;
        break;
    end;
end;
hold off;
```

Program 2: S.H.O.

```
%% Shooting Quantum Fish in a Simple Harmonic Oscillator.
close all; clear all; prec_end = 1; cutoff = .0000001; clc;

set(gcf,'units','normalized','position',[.1,.1,.8,.8]);
set(0,'defaultaxesfontsize',15);
set(0,'defaulttextfontsize',15);
set(0,'defaultlinelinewidth',3);

k = 25;
x = linspace(-2,2,10001); % 21 divisions minimum!
Nx = length(x);
dx = (x(end)-x(1))/(Nx-1);
mid = (Nx-1)/2 + 1;
left = mid - round(1/dx);
right = mid + round(1/dx);
V = .5*k*x.*x; %%  S.H.O.
P(1:Nx) = 0; P(end) = 1; Pswap = P(end);
E = 0;
Ne = 0;
flag2 = 0;
%% Begin search for energy eigenvalues and even
eigenstates:
while 1,
    E = E + 1;
    P(end) = 1;
    while P(end) > cutoff,
        flag2 = 0;
        P(:) = 0;
        P(mid-1:mid+1) = 1;
        for i_P = mid+2:Nx,
            P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))*...
                P(i_P-1)-P(i_P-2);
            P(2*mid-i_P) = P(i_P);
        end;
```

```
        Pswap = P(end);
        prec_end = 1;
        while 1,
            E = E + .1^prec_end;
            P(:) = 0;
            P(mid-1:mid+1) = 1;
            P(end) = Pswap;
            for i_P = mid+2:right-3,
                P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))*...
                    P(i_P-1)-P(i_P-2);
                P(2*mid-i_P) = P(i_P);
            end;
            for i_P = right-2:Nx,
                P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))*...
                    P(i_P-1)-P(i_P-2);
                P(2*mid-i_P) = P(i_P);
            end;
            if abs(P(end)) < cutoff,
                flag2 = 1;
                break;
            elseif Pswap*P(end) < 0
                E = E - 2*(.1^prec_end);
                prec_end = prec_end + 1;
            end;
        end;
        if flag2 == 1,
            break;
        end;
    end;
    N = sqrt(sum(P.*P)*dx);
    P = P/N;
    if Ne == 0,
        plot(x,P,'b-');
        axis([-2,2,-1,11]);
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        title({'Eigenvalues and Eigenfunctions',...
            'for Simple Harmonic Oscillator'},...
            'fontsize',30);
        xlabel('x-position');
        ylabel({'Normalized Eigenfunctions',...
            '(spaced for viewing)'});
        hold on;
        plot(x,V,'k','linewidth',6);
        drawnow;
        Ne = Ne + 2;
    elseif Ne == 2,
        P = P + Ne*2.1;
        plot(x,P,'g-');
```

```
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        drawnow;
        Ne = Ne + 2;
    elseif Ne == 4,
        P = P + Ne*2.1;
        plot(x,P,'m-');
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        drawnow;
        break;
    end;
end;

%% Begin search for energy eigenvalues and odd eigenstates:
cutoff = .000000001;
P(1:Nx) = 0; P(end) = 1; Pswap = P(end);
E = 0;
Ne = 1;
flag2 = 0;
while 1,
    E = E + 1;
    P(end) = 1;
    while P(end) > cutoff,
        flag2 = 0;
        P(:) = 0;
        P(mid) = 0;
        P(mid-1) = -dx;
        P(mid+1) = dx;
        for i_P = mid+2:Nx,
            P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))*...
                P(i_P-1)-P(i_P-2);
            P(2*mid-i_P) = -P(i_P);
        end;
        Pswap = P(end);
        prec_end = 1;
        while 1,
            E = E + .1^prec_end;
            P(:) = 0;
            P(mid) = 0;
            P(mid-1) = -dx;
            P(mid+1) = dx;
            for i_P = mid+2:right-3,
                P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))...
                    *P(i_P-1)-P(i_P-2);
                P(2*mid-i_P) = -P(i_P);
            end;
            for i_P = right-2:Nx,
                P(i_P) = 2*(1-(dx)^2*(E-V(i_P-1)))...
                    *P(i_P-1)-P(i_P-2);
```

```
                P(2*mid-i_P) = -P(i_P);
            end;
            if abs(P(end)) < cutoff,
                flag2 = 1;
                break;
            elseif Pswap*P(end) < 0
                E = E - 2*(.1^prec_end);
                prec_end = prec_end + 1;
            end;
        end;
        if flag2 == 1,
            break;
        end;
    end;
    N = sqrt(sum(P.*P)*dx);
    P = P/N;
    if Ne == 1,
        P = P + Ne*2.1;
        plot(x,P,'r-');
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        hold on;
        drawnow;
        Ne = Ne + 2;
    elseif Ne == 3,
        P = P + Ne*2.1;
        plot(x,P,'c-');
        text(-1.9,Ne*2.1+.3,['E = ',num2str(E,3)]);
        drawnow;
        break;
    end;
end;
hold off;
```

Miscellanea:

A terrific applet showing the radial wavefunction for the user's selected quantum numbers can be found at http://webphysics.davidson.edu/physletprob/ch10_modern/radial.html.

A succinct algorithm for calculating the radial wavefunctions is given at http://quantummechanics.ucsd.edu/ph130a/130_notes/node237.html.