Infinite Square Well Time Evolution

Timothy Holmes

April 27, 2018

The initial step for writing this code was to define a function with the input arguments given by that data file. The input arguments include but are not limited to m as the mass, num as the highest energy eigenvalue, x as a vector of the length of the infinite square well, and $\psi(x,0)$ which is the wave function as an array of numbers. Some other values defined below included $\hbar$ in eV, the full length of the box as L, and an array on n values up to the maximum allowed energy earlier defined as num.

```
function PHY361Homework1(m,num,x,psi0)

%% Constants

hbar = 6.58211951*10^-16;
L = x(2001);
n = 1:num;
```

The first process in before anything can be calculated is to normalize the wave function. This can be done analytically by

$$1 = \int_{-\infty}^{\infty} A^2 |\psi(x)|^2 dx.$$

However the computer can not solve an analytical problem. Instead, this can be done numerically with the trapezoidal method from calculous. Fortunately, matlab has a built in command for this called the trapz command. The trapz command takes an array of numbers that would be considered the bounds and a function of n dimensions by trapz(x,f(x),n). Since $\psi(x,0)$ is complex the complex conjugate of $\psi(x,0)$ needs to be multiplied by $\psi(x,0)$ in the trapz command over the array of length. Then A is solved for as shown below.

```
%% Normalize

A = 1/sqrt(trapz(x,conj(psi0).*psi0));
psi0Norm = A*psi0;
```

The rest of the calculations for this time evolution problem can be done later. Instead setting up plot and various other stuff for the for loop must be done first. When this initial plot it plotted it will plot $\psi(x, 0)$ without its overall phase. The video written is set up to also write a .avi file of the animation after the for loop. Finally, the appropriate time step dt and total time is set up. The time step is important since it determines if $\psi(x, t)$ is going to crawl across the x-axis or zoom across the x-axis. The total time just give the function an appropriate amount of time to run for.

```
%% Plot

fig = figure;
hold on
plotReal = plot(x,real(psi0), 'linewidth', 2);
plotImag = plot(x,imag(psi0), 'linewidth', 2);
plotAbs = plot(x,abs(psi0), 'linewidth', 2);
legend('real','imaginary','Absolute Value')
xlabel('x (nm)')
ylabel('\bf{\psi(t)}')
ylim([-A A]);

video = VideoWriter('TimeEvolution.avi');
open(video);

count = 0;
dt = 50;
timeTotal = 1*10^2;
```

Since $\varphi_n(x)$, c_n, and E_n are all going to be calculated in the for loop they need to be preallocated. This essentially just allows for the computer to have more memory and is efficient when proceeding with large calculations. This also allows for us to think about the vector or matrix sizes of these variables. This comes back to what the input num is. In this case it is 500, therefore we should expect 500 n values. $\varphi_n(x)$ will be a matrix, this is because in needs to have the same length as $\psi(x, 0)$ (as rows) and then 500 columns since the energy eigen state is 500. Thus, we would also have a vector c_n and a vector E_n. Therefore we end up with $\varphi_1(x) \ldots \varphi_{500}(x)$, $c_1 \ldots c_{500}$, and $E_1 \ldots E_{500}$.

```
%% Preallocating

phi = zeros(length(x),num);
c = zeros(1,length(n));
En = zeros(1,length(n));
```

Now that the for loop has come up we can get into the details of calculating the values. $\varphi_n(x)$ for an infinite square well is given by

$$\varphi_n(x) = \sqrt{\frac{2}{L}} * sin\Big(\frac{n\pi x}{L}\Big).$$

The energy eigenstate for an infinite square well is given by

$$E_n = \frac{n^2\pi^2\hbar^2}{2mL^2}.$$

For time evolution the value of c_n still needs to be calculated. This is done analytically by

$$c_n = \int_{-\infty}^{\infty} \varphi_n(x)^* \psi(x,0)dx.$$

However, this again has to be done numerically and is done using the trapz command. This calculation is similar to the previous one in this code. It should be noted that these values can be calculated out side of a for loop. Now that all the essential values are calculated the Schrödinger equation can be solved for by

$$\psi(x,t) = \sum_{n}^{500} c_n e^{-iE_n t/\hbar} \varphi_n(x).$$

The nested for loop calculates for all the values that depend on k. This is really just n, therefore it is calculating $\psi(x,t)$ 500 times and summing it together at the first time value of 0. Then the animation works by using the set command, saying that take the previous plot, update the y values that were just calculated and update the graph. Similarly the video write command does the same. Once it ends it goes to the next time value time + 1 and does the same calculation for all the n values in $\psi(x,t)$. Preallocating $\psi(x,t)$ in the loop is also useful because it removes the previous values for $\psi(x,t)$ and then goes through the loop to calculate them again. This process will continue until it reaches the last time total value previously set.

```
%% Time Evolution Sum

for j = 1:dt:timeTotal

    time = j;
    psi = zeros(size(psi0));

    for k = 1:length(n)

    phi(:,k) = sqrt(2/L)*sin((n(k)*pi.*x)/L);
    En(:,k) = (n(k).^2*pi^2*hbar^2)/(2*m*(L^2));
    c(k) = trapz(x,conj(phi(:,k)).*psi0Norm);
```

```
        psi = psi + c(k).*phi(:,k)*exp(-1i*En(k)*time/hbar);

    end

    count = count + 1;

    title(sprintf('Time Evolution Time Frame Number = %g'
        , time))

    set(plotReal, 'YData', real(psi))
    set(plotImag, 'YData', imag(psi))
    set(plotAbs, 'YData', abs(psi))

    currentFrame = getframe(gcf);
    writeVideo(video,currentFrame);

    drawnow
    pause(0.005)

end

fprintf('Count %f: \n',count)

close(fig);
close(video);

end
```

Below is the .gif that runs the animated wave function (if opened in Adobe Acrobat Reader DC).

The full matlab code with no breaks.

```
%PHY361Homework1
%Time evolution for a continuous system



%% Inputs
% m = mass of particle (an electron in this case)
% num = a scalar that specifies the largest energy eigenstate
% x = a vector that sets the range of the well
% psi0 = a column vector that represents the value of the wave function
% at psi(x,0)
%% Outputs
% A video file as a .avi
% An updated plot through time
%Number of iterations in the for loop
%
%For more detailed description of the code
%please see attached pdf
function PHY361Homework1(m,num,x,psi0)

%% Constants

hbar = 6.58211951*10^-16;
L = x(2001);
n = 1:num;

%% Normalize

A = 1/sqrt(trapz(x,conj(psi0).*psi0)); %normalizing wave function with integratio
psi0Norm = A*psi0; %Normalizing wave function

%% Plot

video = VideoWriter('TimeEvolution.avi'); %starts writting frames
open(video);

count = 0;
dt = 200; %time step
timeTotal = 1*10^10; %total time

%% preallocate

```

```
phi = zeros(length(x),num);
c = zeros(1,length(n));
En = zeros(1,length(n));

%% Time Evolution, eigenstates, c terms, and updates plot

for j = 1:dt:timeTotal

    time = j*1^-18;
    psi = zeros(size(psi0));

    for k = 1:length(n)

    phi(:,k) = sqrt(2/L)*sin((n(k)*pi.*x)/L);
    En(:,k) = (n(k).^2*pi^2*hbar^2)/(2*m*(L^2));
    c(k) = trapz(x,conj(phi(:,k)).*psi0Norm);

    psi = psi + c(k).*phi(:,k)*exp(-1i*En(k)*time/hbar);

    end

    count = count + 1; %checking iteration number

    title(sprintf('Time Evolution Time Frame Number = %g', count))

    mesh(x,x,real(psi.*psi'))
    mesh(x,x,abs(psi.*psi'))

    currentFrame = getframe(gcf); %grabs frams from iteration
    writeVideo(video,currentFrame); %writes frames out to .avi file

    drawnow %draws updated plot
    pause(0.005) %waits for next iteration

end

fprintf('Count %f: \n',count)

close(fig);
close(video);

a = psi0.*psi0';

size(a)

figure(2)
```