Chapter 1

Two-Particle Time-Dependent Scattering (Interaction of Wavepackets)

1.1 Introduction

In August of 2000, an explicit technique was published [1] for the numerical solution of the two-particle, time-dependent Schrödinger equation. This technique was applied to particle-particle scattering when both particles were represented as wavepackets. The results were then rendered as an animated 'movie' for viewing. Although this approach is more complicated than a simple time-independent view of a single particle interacting with a fixed external potential, as found in most quantum mechanics textbooks, it presumably illustrates more closely the physics of nature. As a consequence, students are able to take with them a more lasting impression of this phenomenon.

Here some of the original research codes and additional materials are presented to allow for students to explore this original research topic. These materials have also been provided in a modified form to allow for reasonable results with modest computing power.

1.2 Two-Particle Schrödinger Equation

Animating the interaction of two wavepackets starts with the solution of the two-particle time-dependent Schrödinger equation:

$$i\frac{\partial}{\partial t}\psi(x_1,x_2,t) = H\psi(x_1,x_2,t), \tag{1.1}$$

$$H = -\frac{1}{2m_1}\frac{\partial^2}{\partial x_1^2} - \frac{1}{2m_2}\frac{\partial^2}{\partial x_2^2} + V(x_1,x_2). \tag{1.2}$$

Here, for simplicity, we assume a one-dimensional space where $\hbar = 1$ and take m_i and x_i to be the mass and position of particle $i = 1, 2$. Knowledge of the two-particle wave function $\psi(x_1, x_2, t)$ permits the calculation of the joint probability density for particle 1 being at x_1 and particle 2 being at x_2 at time t:

$$\rho(x_1,x_2,t) = \left|\psi(x_1,x_2,t)\right|^2. \tag{1.3}$$

The fact that each particles must be located someplace in space leads to the normalization constraint:

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} dx_1\,dx_2\,\left|\psi(x_1,x_2,t)\right|^2 = 1. \tag{1.4}$$

If particles 1 and 2 are identical, then their total wave function should be symmetric or antisymmetric under interchange of the particles. This condition may be imposed on the numerical solution $\psi(x_1, x_2)$, by forming the combinations

$$\psi^{'}(x_1, x_2) = \frac{1}{\sqrt{2}} \left[\psi(x_1, x_2) \pm \psi(x_2, x_1)\right] \quad \Longrightarrow \tag{1.5}$$

$$2\rho(x_1, x_2) = |\psi(x_1, x_2)|^2 + |\psi(x_2, x_1)|^2 \pm 2\mathrm{Re}\left[\psi^*(x_1, x_2)\psi(x_2, x_1)\right]. \tag{1.6}$$

While this wave function is the true solution, it is difficult, at best, to 'picture' the physics in a three-variable complex function and a means for extracting information leading to the visualization of two separate colliding wavepackets is desirable.

Under circumstances where the hamiltonian is separable, the two particle wave function takes on a product form:

$$H(x_1, x_2) = H_1(x_1) + H_2(x_2) \Longrightarrow \psi(x_1, x_2) = \psi_1(x_1)\psi_2(x_2), \tag{1.7}$$

allowing for the immediate retrieval of each particle's density function. For interacting wavepackets, however, the potential term in equation (1.2) is a function of the distance between particles, and having $|x_2 - x_1|$ as an independent variable does not allow for its separation. All is not lost though, for meaningful one particle probability densities may be obtained by integrating out the position dependence of the other particle:

$$\rho_i(x_i, t) = \int_{-\infty}^{+\infty} dx_j \, \rho(x_1, x_2, t), \quad (i \neq j = 1, 2). \tag{1.8}$$

1.3 Numerical Method

The two-particle Schrödinger equation (1.1) is solved via a finite difference method that converts the partial differential equation into a set of simultaneous, algebraic ones. First, the dependent variable ψ on a grid of discrete values for the independent variables is evaluated:

$$\psi(x_1, x_2, t) = \psi(x_1 = l\Delta x_1, x_2 = m\Delta x_2, t = n\Delta t) \equiv \psi^n_{l,m}, \tag{1.9}$$

where l, m, and n are integers. The space part of the algorithm is based on Taylor expansions of $\psi(x_1, x_2, t)$ in *both* the x_1 and x_2 variables up to $\mathcal{O}(\Delta x^4)$; for example,

$$\frac{\partial^2 \psi}{\partial x_1^2} \simeq \frac{\psi(x_1 + \Delta x_1, x_2) - 2\psi(x_1, x_2) + \psi(x_1 - \Delta x_1, x_2)}{\Delta x_1^2} + \mathcal{O}(\Delta x_1^2). \tag{1.10}$$

In discrete notation, the RHS of the Schrödinger equation (1.1) now becomes:

$$H\psi = -\frac{\psi_{l+1,m} - 2\psi_{l,m} + \psi_{l-1,m}}{2m_1\Delta x_1^2} - \frac{\psi_{l,m+1} - 2\psi_{l,m} + \psi_{l,m-1}}{2m_2\Delta x_2^2} + V_{lm}\psi_{l,m}. \tag{1.11}$$

Next, the time derivative in (1.1) must be expressed in terms of finite time differences. Using the formal solution to the time-dependent Schrödinger equation, consider initially making a forward-difference approximation for time evolution operator. Remembering the fact that the exponential of the operator, H, is itself an operator defined in terms of its Taylor series expansion it follows that:

$$\psi^{n+1}_{l,m} = e^{-i\Delta t H}\psi^n_{l,m} \simeq (1 - i\Delta t H - \Delta t^2 H^2 + \cdots)\psi^n_{l,m}. \tag{1.12}$$

Although simple, this approximation scheme is unstable. Ignoring terms above $\mathcal{O}(\Delta t)$, the expression multiplying ψ now has eigenvalue $(1 - iE\Delta t)$ and modulus $\sqrt{1 + E^2\Delta t^2}$. This means the modulus of the wave function will increase with each time step [2]. An improvement introduced by Askar and

Cakmak [3] is a central difference algorithm also based on the formal solution (1.12):

$$\psi_{l,m}^{n+1} - \psi_{l,m}^{n-1} = \left(e^{-i\Delta t H} - e^{i\Delta t H}\right)\psi_{l,m}^{n} \simeq -2i\Delta t H \psi_{l,m}^{n}, \tag{1.13}$$

$$\Rightarrow \quad \psi_{l,m}^{n+1} \simeq \psi_{l,m}^{n-1} - 2i\left[\left\{(\frac{1}{m_1}+\frac{1}{m_2})4\lambda + \Delta t V_{l,m}\right\}\psi_{l,m}^{n} - \lambda\left\{\frac{1}{m_1}(\psi_{l+l,m}^{n} + \psi_{l-1,m}^{n}) + \frac{1}{m_2}(\psi_{l,m+1}^{n} + \psi_{l,m-1}^{n})\right\}\right], \tag{1.14}$$

where it is assumed $\Delta x_1 = \Delta x_2$ and the ratio $\lambda = \Delta t/\Delta x^2$ is formed.

Equation (1.14) is an *explicit* solution in which the wave function at only two past time values must be stored simultaneously in memory to determine all future times by continued iteration. In contrast, an *implicit* solution determines the wave function for all future times in just one step, yet this one step requires the solution of simultaneous algebraic equations involving all space and time values. Accordingly, an implicit solution requires the inversion of exceedingly large matrices.

While the explicit method (1.14) produces a solution which is stable and second-order accurate in time, in practice, it does not conserve probability well. Visscher [4] has deduced an improvement which takes advantage of the extra degree of freedom provided by the complexity of the wave function to preserve probability better. If the wave function is separated into real and imaginary parts,

$$\psi_{l,m}^{n+1} = u_{l,m}^{n+1} + i\, v_{l,m}^{n+1}, \tag{1.15}$$

the algorithm (1.14) separates into the pair of coupled equations:

$$u_{l,m}^{n+1} = u_{l,m}^{n-1} + 2\left[\left\{(\frac{1}{m_1}+\frac{1}{m_2})4\lambda + \Delta t V_{l,m}\right\} v_{l,m}^{n} - \lambda\left\{\frac{1}{m_1}(v_{l+1,m}^{n} + v_{l-1,m}^{n}) + \frac{1}{m_2}(v_{l,m+1}^{n} + v_{l,m-1}^{n})\right\}\right], \tag{1.16}$$

$$v_{l,m}^{n+1} = v_{l,m}^{n-1} - 2\left[\left\{(\frac{1}{m_1}+\frac{1}{m_2})4\lambda + \Delta t V_{l,m}\right\} u_{l,m}^{n} - \lambda\left\{\frac{1}{m_1}(u_{l+1,m}^{n} + u_{l-1,m}^{n}) + \frac{1}{m_2}(u_{l,m+1}^{n} + u_{l,m-1}^{n})\right\}\right]. \tag{1.17}$$

Visscher's advance evaluates the real and imaginary parts of the wave function at slightly different (staggered) times,

$$[u_{l,m}^{n},\, v_{l,m}^{n}] = [\mathrm{Re}\,\psi(x,t),\, \mathrm{Im}\,\psi(x,t+\frac{1}{2}\Delta t)], \tag{1.18}$$

and uses a definition for probability density that differs for integer and half-integer time steps,

$$\rho(x,t) = |\mathrm{Re}\,\psi(x,t)|^2 + \mathrm{Im}\,\psi(x,t+\frac{\Delta t}{2})\,\mathrm{Im}\,\psi(x,t-\frac{\Delta t}{2}), \tag{1.19}$$

$$\rho(x,t+\frac{\Delta t}{2}) = \mathrm{Re}\,\psi(x,t+\Delta t)\mathrm{Re}\,\psi(x,t) + \left|\mathrm{Im}\,\psi(x,t+\frac{\Delta t}{2})\right|^2. \tag{1.20}$$

These definitions reduce to the standard one for infinitesimal Δt, and provide an additional algebraic cancellation of errors so that probability is better conserved.

1.4 Initial and Boundary Conditions

Two interaction potentials between particles are provided within the accompanying codes; a 'soft' potential with a Gaussian dependence, and a 'hard' one with a square-well dependence, both with range α and strength V_0:

$$V(x_1,x_2) = \begin{cases} V_0 \exp[-\frac{|x_1-x_2|^2}{2\alpha^2}] & \text{(Gaussian)} \\ V_0\ \theta(\alpha - |x_1-x_2|) & \text{(Square)} \end{cases}. \tag{1.21}$$

In both situations the initial placement of the two particles is represented by a product of individual one-particle wavepackets:

$$\psi(x_1, x_2, t = 0) = e^{ik_1 x_1} \exp[-\frac{(x_1 - x_1^0)^2}{4\sigma^2}] \times e^{ik_2 x_2} \exp[-\frac{(x_2 - x_2^0)^2}{4\sigma^2}]. \tag{1.22}$$

where particle 1 resides at x_1^0 and particle 2 at x_2^0. Because of the Gaussian factors, ψ is not an eigenstate of the particle i momentum operators $-i\partial/\partial x_i$, but instead contains a spread of momenta about the mean, initial momenta k_1 and k_2. Also note that if the wavepacket is made very broad ($\sigma \to \infty$), momentum eigenstates are obtained.

The staggered-time algorithm is started with the real part the wave function (1.22) at $t = 0$ and the imaginary part at $t = \Delta t/2$. The initial imaginary part follows by assuming that $\Delta t/2$ is small enough, and σ large enough, for the initial time dependence of the wavepacket to be that of the plane wave parts:

$$\begin{aligned} \mathrm{Im}\,\psi(x_1, x_2, t = \frac{\Delta t}{2}) \quad &\simeq \quad \sin\left[k_1 x_1 + k_2 x_2 - \left(\frac{k_1^2}{2m_1} + \frac{k_2^2}{2m_2}\right)\frac{\Delta t}{2}\right] \\ &\quad \times \exp - \left[\frac{(x_1 - x_1^0)^2 + (x_2 - x_2^0)^2}{2\sigma^2}\right]. \end{aligned} \tag{1.23}$$

In a scattering experiment, the projectile enters from infinity and the scattered particles are observed at infinity. This is modeled by solving the partial differential equation within a box of side L (ideally) much larger than both the range of the potential and the width of the initial wavepacket. This leads to the boundary conditions:

$$\psi(0, x_2, t) = \psi(x_1, 0, t) = \psi(L, x_2, t) = \psi(x_1, L, t) = 0. \tag{1.24}$$

The largeness of the box minimizes the effects of the boundary conditions during the collision of the wavepackets, although at large times there will be interesting, yet artificial, collisions with the box.

1.5 Simulations

The materials are presently provided to run these research codes within the Unix environment. Work is currently underway to make these tools available for Linux.

1.5.1 Codes

Two programs are provided coded in the C language. The one containing the Gaussian potential of equation (1.21) is `packets.c` and the other, with the square dependence, is `square.c`. The `square.c` code has been trimmed to a more streamlined form in which it is able to run within more reasonable time and memory constraints. This is the code that will be focused on. The other, `packets.c`, is a full-blown research code requiring significant resources to run.

1.5.2 Coded Parameters

The programs contain two, user defined, variables whose values are encoded within the source files. These values may be changed with a text editor prior to compilation.

The first such variable is `xDim` and defines how many space intervals exist across our one-dimensional box of size L (1.24). Care must be taken to ensure this value is odd as required by the integration algorithm used within the code.

The second is `choice` which may take on any one of the integer values: {0,1,2,3}. Setting `choice`=0 will output the joint probability density function as given in (1.3). To view separate colliding wavepackets as determined by (1.8) leave the default value of `choice`=1. Choosing 2 or 3 will yield information on the correlation or the superposition of the individual density functions respectively.

1.5.3 File Read Parameters

A file called `in.dat` (Unix is case sensitive) must exist within the same directory as the executable. This is a text file containing the additional parameters, needed by the code, to be read in during runtime. A typical `in.dat` file would look like:

```
2000
110.
110.
0.005
4.2E-06
0.05
0.25
0.60
0.5
5.
+90000.
0.062
100
0
```

The entries represent (in order):

1. `Nt` (integer). The number of total time steps to be taken before termination of the run.
2. `k1` (float). k_1 in equation (1.22). Kinetic energy for m_1 is about $k^2/2m_1$.
3. `k2` (float). -k_2 in equation (1.22). This wave vector, for packet 2, is given as positive but interpreted as negative by the code as to yield oppositely traveling (colliding) packets.
4. `dx` (float). The size of each space step to be taken. $\Delta x_1 = \Delta x_2 = \Delta x$ in (1.9). L in (1.24) is taken by the code to be unity so the number of steps `xDim` and the space step `dx` are related by `xDim` $*$ `dx` $= 1.0$.
5. `dt` (float). The time step Δt in (1.9). Relatively large time steps must be avoided for when the product, `dt*vmax`, of the time step and the potential exceeds 0.38 the algorithm becomes unstable and the wavefunction computed from one time step to the next blows up losing its physical meaning.
6. `sig` (float). This is σ in (1.22), the wavepacket width parameter.
7. `x01` (float). Initial position of wavepacket for particle 1, x_1^0 in (1.22).
8. `x02` (float). x_2^0 in (1.22).
9. `m1` (float). Mass of particle 1, m_1 in (1.2).
10. `m2` (float). m_2 in (1.2).
11. `vmax` (float). Strength parameter for the potential, V_0 in (1.21). It can be positive (for repulsive potentials) or negative (for attractive potentials). As indicated earlier, if you change this value, be sure that `dt*vmax` is less than or equal to 0.38 since these two parameters appear as a product in the equations to be solved (refpsireal).
12. `alpha` (float). This is α in (1.21). It is a potential width parameter.
13. `nprint` (integer). This is a multiple of the time step for which the program dumps data to output files. Selecting 100, for example, means that there will be output data files for time steps 1, 100, 200,... up to the maximum value `Nt`.

14. `syymetry` (integer). A flag that assumes one of three integer values, {-1,0,1}, and imposes a symmetry condition on the wave function (see equation (1.6)).

    - -1: antisymmetrization.
    - 0: no symmetrization.
    - +1: symmetrization.

1.5.4 Execution

Within the Unix environment compiling the source code is done by issuing the command: "`gcc -lm square.c -o square`". Additional help may be obtained by entering: "`man gcc`". Running the code is now carried out by simply typing "`./square`". As the program runs, a series of data files will be written to the current directory. These files will appear as `run.00001, run.00100, run.00200,...` (where the displayed enumeration here is based on `nprint`=100). For `choice`=1, each of these data files will consist of three columns:

1. (integer) Specifies the position within the one dimensional box. Here zero scales to 0 and `xDim` scales to L (taken to be 1).
2. (float) Gives the relative probability of particle 1 being found at this position for the time corresponding to the file.
3. (float) Gives the relative probability of particle 2 being found at this position for the time corresponding to the file.

At the termination of the programs execution, the output data files may be processed for visualization.

1.6 Data Processing

The second and third column are both ordinates that share the same abscissa (first column). The two plots should be overlaid on the same graph, using a favorite graphing package, so the interaction between wavepackets is made apparent. A popular such package is Gnuplot. Accomplishing the above task can be done by entering the command:

```
plot 'run.00100' using 1:2 w l, plot 'run.00100' using 1:3 w l
```

For more instruction or specific formatting commands type "`man gnuplot`".

As each of the output files is graphed, the images should be saved as `.gif` files. Animation is accomplished by the rapid, ordered viewing of these `.gif` files in succession. Gnuplot itself cannot save to `.gif` format so intermediate formats (such as pixmap) become necessary as well as additional programs that will open them. The last one of course needs to be able to output `.gif`. One is now faced with the repetitive task of opening and saving a large number of similar files with a number of different programs. This sounds tedious and it sure would be nice if there was a convenient way of doing it!

1.6.1 Script File

To this end a script can be formed to automate the job. A script is a file containing a series of commands within control structures capable of being run within the shell to perform tedious tasks. Such a script is provided and is called `mscript`, a portion of it appears as:

```
#! /bin/ksh
unalias rm
integer i=$1
while test i -lt $2 do
```

```
if test i -lt $2
 then
 print "set terminal pbm small color;
 set output\"$3t=0000$i.ppm\";
 set noxtics; set noytics;
 set size 1.0, 1.0; set yrange [0:.1];
 plot 'run.0000$i' using 1:2 w lines, 'run.0000$i' using 1:3 w lines;
 " >data0000$i.gnu
 gnuplot data0000$i.gnu
 ppmquant -map samp $3t=0000$i.ppm>$3at=0000$i.ppm
 ppmtogif -map samp $3at=0000$i.ppm > $30000$i.gif
 rm $3t=0000$i.ppm $3at=0000$i.ppm data0000$i.gnu
 i=i+99
 fi
```

⋮

- The #! line lets the computer know to carry out the subsequent commands in the korn shell.
- The symbol $i indicates that i is a variable.
- The symbol > pipes output to the following file.
- The ppmquant command takes a pixmap and maps an exsisting set of colors to new ones. This is why the map file `samp` must exsist within the directory.
- The counter increments at the bottom of the loops must be set by the user to coincide with the choice of `nprint`. The first counter should increment by the value `nprint-1` (This is because the first file written is `.00001` not `.00000`.) and the rest by `nprint`.

The script file, along with the file `samp`, should be placed in the directory containing the `run.` output files. The script is launched by entering: "`./mscript 1 Nt+1 name`" where `Nt+1` is replaced by its value and `name` is replaced by the users choice of output filename. It takes about 10 minutes to run and upon successful completion of the script, there should be a new set of `.gif` files bearing the users chosen `name` followed by the familiar `nprint` multiples.

1.6.2 Animation

The final act of merging the individual `.gif` files into a `.gif` movie is a step that has been taken out of the script. This is because the required utility, Gifmerge, is only currently available on two machines: goophy and daphy.physics.orst.edu. Gifmerge is called upon by typing: "`gifmerge -10 *.gif>movie`". Here the `-10` separates the frames by 0.1 sec. in time and the * refers to the `name` given above. The output `movie.gif` may be opened as a file and viewed on a standard internet browser.

1.7 Tuning and Exploration

After the algorithm has been decided upon and the code written comes what is often considered by computational physicists as the most difficult task. As was hinted at earlier, the method adopted here has a window of stability. This is to say that not all possible values for parameters, nor all time durations (`Nt*dt`), will yield accurate predictions. Indeed no experiment, even the best ones, can bear precise answers outside of its design. If one were to measure the speed of a snail and a bullet using only a meter stick and a fine Swiss watch without a second hand, which of the results could be expected to hold any merit? It becomes important to know ones 'tools' and where they are applicable.

1.7.1 Time Steps

It was stated above that when the product, `dt*vmax`, of the time step and the potential exceeds 0.38, the algorithm becomes unstable and the wavefunction computed from one time step to the next blows up. Verify this assertion by running `square` successively with gradually increasing time steps such that the product of 0.38 is exceeded. As the time steps are increased, `Nt` and `nprint` should also be adjusted so that the time duration (`Nt*dt`) and the number of output files is approximately the same.

The normalization (For the code `square.c`, particular constants have been dropped which is why ρ_i (1.8) is not normalized to unity.) of the output wavepackets should be initially fixed. As the product 0.38 is exceeded, the area under the packets should increase both with subsequent files and as the time step is incremented. Finally the output will surpass what the machine can represent and overflow errors will occur. What could be expected if the time step was nearly stable (a product close to 0.38) and the code allowed to run for a long time duration (`Nt*dt`)?

1.7.2 Space Steps

Making the space step `dx` to large leads to spurious ripples during interactions. Particularly for soft potentials, a 'spiked' interaction would not be expected. Earlier publications have claimed this effect to be physical but it was shown later to be only computational instability. Experiment with this parameter and see if a range of acceptable values can be established.

1.7.3 Other Criteria

The algorithm used here was optimized by using Visscher's advance to better preserve probability. Could another preserved quantity, such as energy, have been used to optimize the algorithm? What benefits or drawbacks could be expected in doing this? Experiment with other parameters within the code to gain insight into the physics as well as the limitations of this research topic. Additional materials can also be found on the web [5].

Bibliography

[1] R.H. Landau, J. Maestri and M.J. Páez, *Two-Particle Schrödinger Equation Animations of Wavepacket-Wavepacket Scattering,* Amer. J. Phys., **68**, 000–000 (2000).

[2] S.E. Koonin, *Computational Physics,* Benjamin, Menlo Park, 176–178 (1986).

[3] A. Askar and A.S. Cakmak, *Explicit Integration Method for the Time-Dependent Schrödinger Equation for Collision Problems,* J. Chem. Phys., **68**, 2794–2798 (1978).

[4] P.B. Visscher, *A Fast Explicit Algorithm for the Time-Dependent Schrödinger Equation,* Computers In Physics, 596–598 (Nov/Dec 1991).

[5] *Movies of Wavepacket-Wavepacket Quantum Scattering,* `href://nacphy.physics.orst.edu/ComPhys/PACKETS/`