Solving the Time-Dependent Schrödinger Equation

Blog version

Paul Durham

Scientific Computing Department, STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, UK

2 June 2020

Abstract

Notes on the numerical solution of the time-dependent Schrödinger equation (TDSE). The structure of the 1-dimensional code **TDSE1** is described. The results of tests against analytic, semi-analytic and alternative numerical approaches are given. A slightly modified code **ITP** for finding bound states by imaginary time propagation is also described

Contents

1 Introduction

My C++ code, **TDSE1**, solves the 1-electron time-dependent Schrödinger equation (TDSE) in 1 dimension, $\mathbb{R}^1$, by treating it as a partial differential equation (PDE) described on a mesh of points in space (x) and time (t). There are no basis functions, eigenstates or anything like that – this is not a spectral method. The wave function at any point and time is thus represented by a complex number and the set of such numbers form wave function vectors.

I think of TDSE1 as a kind of toy code, because I designed for interesting toy problems which can't be solved analytically even in $\mathbb{R}^1$. It was also an exploratory project for me, learning about the numerical solution of PDEs, rather than the ordinary differential equations usually encountered in electronic structure calculations. I wanted to get into the nuts and bolts of solving time-dependent quantum problems numerically.

The problems to be handled with this code are initial value problems: the initial state (in this case the wave function at time t_0) is given as a set of values (ie a complex vector) on the x-mesh, and the code then finds the wave function vector on the same x-mesh at some later time. The problems for which this method is generally essential are those in which the Hamiltonian is time-dependent – driven systems. However, for testing purposes we use it in these notes (sections 4.1 and 4.2) to study some problems in which the Hamiltonian is time-independent. Finally, in section 5, we describe a modification in which a fictitious propagation through imaginary time can be used to find bound states.

1.1 The x-t mesh

We use a regular space mesh in the interval $x_{\min} \le x \le x_{\max}$ given by the set of N_x points

$$x_j = x_{\min} + j\Delta x, \quad j = 0,1,2,\ldots,N_x - 1 \tag{1.1}$$

where

$$\Delta x = \frac{x_{\max} - x_{\min}}{N_x - 1} \tag{1.2}$$

The code propagates the wave function vector forward from an initial time t_0 in N_t steps of duration Δt to a final time $t_0 + (N_t - 1)\Delta t$. Note that there is a relationship between the x-mesh increment Δx and the time δt through which one can expect to propagate the solution with stability and with a given accuracy – see [1] and [2]. But Δt just specifies the times at which one requires the wave function. Often it will be the case that $\Delta t > \delta t$. Thus within the driver function **Propagate**, the time increment Δt is divided, if necessary, into a number of intermediate steps of duration δt chosen to ensure stability and accuracy, and the propagation is done sequentially through these intermediate steps.

2 The Cayley-Crank-Nicolson Algorithm

We start from the usual form for time–propagation involving the evolution operator U (we work in the position representation but suppress variable x for simplicity of notation):

$$\psi(x,t+\delta t)\rightarrow\psi(t+\delta t)=U(t+\delta t,t)\psi(t) \tag{2.1}$$

Here the equation of motion of the evolution operator is driven by the Hamiltonian $H(t)$:

$$i\frac{\partial}{\partial t}U(t,t_0)=H(t)U(t,t_0) \tag{2.2}$$

for which the formal solution is [3]

$$U(t,t_0)=1-i\int_{t_0}^{t}dt_1H(t_1)U(t_1,t_0) \tag{2.3}$$

Splitting the interval δt in half and applying the composition property of the evolution operator [4] we can write (2.1) in the following equivalent ways:

$$\begin{aligned}\psi(t+\delta t)&=U(t+\delta t,t+\delta t/2)U(t+\delta t/2,t)\psi(t)\\ U^{-1}(t+\delta t,t+\delta t/2)\psi(t+\delta t)&=U(t+\delta t/2,t)\psi(t)\end{aligned} \tag{2.4}$$

Since the evolution operator is unitary, we therefore have

$$U^{\dagger}(t+\delta t,t+\delta t/2)\psi(t+\delta t)=U(t+\delta t/2,t)\psi(t) \tag{2.5}$$

So far, so exact. But if we iterate (2.3) in the usual way, we get

$$\begin{aligned}U(t+\delta t/2,t)&=1-i\int_{t}^{t+\delta t/2}dt'H(t')+O\left((\delta t)^2\right)\\ U^{\dagger}(t+\delta t,t+\delta t/2)&=1+i\int_{t+\delta t/2}^{t+\delta t}dt'H(t')+O\left((\delta t)^2\right)\end{aligned} \tag{2.6}$$

Therefore we can approximate (2.5) as follows:

$$\left(1+i\int_{t+\delta t/2}^{t+\delta t}dt'H(t')\right)\psi(t+\delta t)\approx\left(1-i\int_{t}^{t+\delta t/2}dt'H(t')\right)\psi(t) \tag{2.7}$$

This is the so-called Cayley form [1] of the Crank-Nicolson algorithm. Note that the rhs looks like an explicit scheme whereas the lhs looks implicit – it's a mixture of the two schemes. As demonstrated in sections 2.1 and 2.2, this form is 2nd order accurate in δt and maintains unitarity (and hence the normalisation of the wave function). If we approximate the action of the Hamiltonian in space by the 2nd order difference scheme described in section 2.3, we obtain the Crank-Nicolson algorithm for the TDSE – see [1] and [2]. Hence this algorithm is 2nd order accurate in time, unitary and unconditionally stable.

2.1 Accuracy in the time-step

From (2.1) and (2.3) we can write

$$\psi(t+\delta t)=U(t+\delta t,t)\psi(t)=\left(1-i\int_{t}^{t+\delta t}dt_1H(t_1)U(t_1,t)\right)\psi(t) \tag{2.8}$$

Iterating this leads to

$$\begin{aligned}\psi(t+\delta t) &= \left(1-i\int_t^{t+\delta t} dt_1 H(t_1)\left(1-i\int_t^{t_1} dt_2 H(t_2)U(t_2,t)\right)\right)\psi(t) \\ &= \left(1-i\int_t^{t+\delta t} dt_1 H(t_1)\left(1-i\int_t^{t_1} dt_2 H(t_2)\right)\right)\psi(t)+O\left((\delta t)^3\right) \\ &= \left(1-i\int_t^{t+\delta t} dt_1 H(t_1)-\int_t^{t+\delta t} dt_1\int_t^{t_1} dt_2 H(t_1)H(t_2)\right)\psi(t)+O\left((\delta t)^3\right)\end{aligned} \tag{2.9}$$

Up to this point, all the time-ordering in the integrals is correct and no assumptions have been made about the Hamiltonian commuting with itself at different times. But now let's suppose that δt is small compared with the time-scale on which the fractional change in the Hamiltonian is significant:

$$\frac{\delta H}{H} \sim \frac{\delta t}{H}\left(\frac{\partial H}{\partial t}\right) \ll 1 \tag{2.10}$$

If this holds we can treat the Hamiltonian as a constant. Then we have

$$\psi(t+\delta t) = \left(1-iH\delta t-\frac{1}{2}H^2(\delta t)^2\right)\psi(t)+O\left((\delta t)^3\right) \tag{2.11}$$

It is easy to verify that for a constant Hamiltonian the Cayley-Crank-Nicolson (CCN) form (2.7) ie

$$(1+iH\delta t/2)\psi(t+\delta t) = (1-iH\delta t/2)\psi(t)+O\left((\delta t)^2\right) \tag{2.12}$$

gives exactly this answer up to 2nd order in δt, and therefore it is of 2nd order accuracy in the time-step. Note that for this demonstration to work for a general time-dependent Hamiltonian[1] the condition (2.10) is necessary because of the time-ordering of the integrals.

2.2 Unitarity

Consider the CCN approximation (2.7) and Taylor expand the Hamiltonian about time $t+\delta t/2$:

$$H(t') = H+(t'-t-\delta t/2)H'+O\left((\delta t)^2\right) \tag{2.13}$$

where $H \equiv H(t+\delta t/2)$, $H' \equiv \left.\frac{dH}{dt}\right|_{t+\delta t/2}$. Using this expansion on both sides, we can easily see that (2.7) reduces to (2.12) with the second order error term being proportional to H'.

To demonstrate that this CCN approximation is unitary, write

$$\begin{aligned}\psi(t+\delta t) &= (1+iH\delta t/2)^{-1}(1-iH\delta t/2)\psi(t)+O\left((\delta t)^2\right) \\ &\equiv U_C(t+\delta t,t)\psi(t)+O\left((\delta t)^2\right)\end{aligned} \tag{2.14}$$

Because H is Hermitian we can write

$$U^{\dagger}{}_C(t+\delta t,t) = (1-iH\delta t/2)^{-1}(1+iH\delta t/2) \tag{2.15}$$

and so the Cayley form U_C is obviously unitary:

[1] "General" here means that the Hamiltonians at different times may not commute – see [4].

$$U^{\dagger}{}_{C}\left(t+\delta t,t\right)U_{C}\left(t+\delta t,t\right)=1 \tag{2.16}$$

Note: the key step here is that the Hamiltonians on both sides of (2.7) are Taylor expanded about the same time $t+\delta t/2$. If they weren't then the exactness of (2.16) would have to rely on the commutation on the Hamiltonian at different times, and we can't assume that. But for practical calculations with time-dependent Hamiltonians, this section's message is that in implementing the CCN algorithm one should evaluate both Hamiltonian terms at time $t+\delta t/2$.

2.3 The differencing scheme

We imagine evaluating the wave function on the real-space mesh described in section 1.1, and denote these values by the vector $\mathbf{y}=\left\{y_j\left(t\right),\ j=0,\ldots,N_x-1\right\}$

$$\psi\left(x_j t\right)\equiv y_j\left(t\right) \tag{2.17}$$

The Hamiltonian in atomic units is

$$H\left(x,t\right)=-\frac{1}{2}\frac{\partial^2}{\partial x^2}+V\left(x,t\right) \tag{2.18}$$

When this operates on the wave function, we need to work out the 2nd derivatives of the wave function at the x-mesh points, and we use the standard 2nd order formula for this:

$$\begin{aligned}\left.\frac{\partial^2\psi\left(x,t\right)}{\partial x^2}\right|_{x=x_j} &\approx \frac{\psi\left(x_j-\Delta x,t\right)-2\psi\left(x_j,t\right)+\psi\left(x_j+\Delta x,t\right)}{\left(\Delta x\right)^2}\\ &=\frac{y_{j-1}\left(t\right)-2y_j\left(t\right)+y_{j+1}\left(t\right)}{\left(\Delta x\right)^2}\end{aligned} \tag{2.19}$$

According to section 2.2 we need to find the action of $H\left(t+\delta t/2\right)$ on the wave function $\psi\left(t\right)$. For the above differencing scheme this is

$$H\left(x,t+\delta t/2\right)\psi\left(x,t\right)=-\frac{1}{2\left(\Delta x\right)^2}\left(y_{j-1}\left(t\right)-2y_j\left(t\right)+y_{j+1}\left(t\right)\right)+V_j\left(t+\delta t/2\right)y_j\left(t\right) \tag{2.20}$$

Which can be represented by a matrix multiplication $\mathbf{H}(t+\delta t/2)\mathbf{y}\left(t\right)$.

The matrix $\mathbf{H}$ thus has a tridiagonal structure illustrated below for $N_x=6$:

$$\mathbf{H}\left(t\right)=\begin{pmatrix} d_0 & u_0 & 0 & 0 & 0 & 0\\ l_1 & d_1 & u_1 & 0 & 0 & 0\\ 0 & l_2 & d_2 & u_2 & 0 & 0\\ 0 & 0 & l_3 & d_3 & u_3 & 0\\ 0 & 0 & 0 & l_4 & d_4 & u_4\\ 0 & 0 & 0 & 0 & l_5 & d_5 \end{pmatrix} \tag{2.21}$$

From (2.20) we see that the matrix can be specified by the three vectors: sub-diagonal $\left(\mathbf{l}\right)$, diagonal $\left(\mathbf{d}\right)$ and super-diagonal $\left(\mathbf{u}\right)$

$$\mathbf{l}(t) = -\frac{1}{2(\Delta x)^2}\{1,1,\ldots,1\} = -\frac{\hat{\mathbf{e}}}{2(\Delta x)^2}$$
$$\mathbf{d}(t) = \left\{V_0(t) + \frac{1}{(\Delta x)^2}, V_1(t) + \frac{1}{(\Delta x)^2}, \ldots, V_{N-1}(t) + \frac{1}{(\Delta x)^2}\right\} = \mathbf{V}(t) + \frac{\hat{\mathbf{e}}}{(\Delta x)^2} \quad (2.22)$$
$$\mathbf{u}(t) = -\frac{1}{2(\Delta x)^2}\{1,1,\ldots,1\} = -\frac{\hat{\mathbf{e}}}{2(\Delta x)^2}$$

where $V_j(t) \equiv V(x_j, t)$ and $\mathbf{V}(t) = \{V_0(t), V_1(t), \ldots, V_{N_x-1}(t)\}$. Here $\hat{\mathbf{e}}$ is the N_x dimensional unit vector. Note that elements l_0 and u_{N_x-1} are absent from the matrix **H** - see (2.21) for example – and are therefore never referenced in algorithms.

Now we see from (2.14) that we can write the CCN approximation as follows:

$$\left(1 + iH(t + \delta t/2)\delta t/2\right)\psi(t + \delta t) = \left(1 - iH(t + \delta t/2)\delta t/2\right)\psi(t) + O\left((\delta t)^2\right) \quad (2.23)$$

We now know how the Hamiltonian operates on the wave functions. If we define tridiagonal matrices

$$\mathbf{P}^{(+)}(t, \delta t/2) \equiv 1 - \frac{i\delta t}{2}\mathbf{H}(t + \delta t/2)$$
$$\mathbf{P}^{(-)}(t, \delta t/2) \equiv 1 + \frac{i\delta t}{2}\mathbf{H}(t + \delta t/2) \quad (2.24)$$

then (2.23) becomes

$$\mathbf{P}^{(-)}(t, \delta t/2)\mathbf{y}(t + \delta t) = \mathbf{P}^{(+)}(t, \delta t/2)\mathbf{y}(t) + O\left((\delta t)^2\right) \quad (2.25)$$

The notation here reflects the fact that $\mathbf{P}^{(+)}$ represents propagation forward in time while $\mathbf{P}^{(-)}$ represents propagation "backward" in time.

(2.25) is a set of linear equations for the components of the propagated wave function vector $\mathbf{y}(t + \delta t)$ in terms of the initial wave function vector $\mathbf{y}(t)$. This tridiagonal system is set up and solved in function **Prop_CCN**.

3 The structure of the code

The initial wave function is set up at the beginning of a run by calling function **Yzero**, which, for each point in the x-mesh, calls a function to evaluate the chosen wave function at any point x. This can take many different forms; in the present version of the code the function called by **Yzero** is **Gauss**, which simply returns the normalised Gaussian function of width l, multiplied by a phase factor corresponding to a wave-packet moving with wave vector (ie speed in au) k_0, namely

$$\psi_0(x) = \left(l\sqrt{\pi}\right)^{-1/2} e^{-x^2/2l^2} e^{ik_0 x} \quad (3.1)$$

3.1 Function **Prop_CCN**

As noted in section 2.3, **Prop_CCN** solves the tridiagonal system in equation (2.25). First, the three vectors (subdiagonal, diagonal and superdiagonal) comprising the tridiagonal matrices $\mathbf{P}^{(+)}$ and $\mathbf{P}^{(-)}$ are found. From (2.24) and (2.22) we find

$$\begin{aligned}
\mathbf{p}_l^{(+)}(t,\delta t/2) &= \frac{i\delta t}{4(\Delta x)^2}\hat{\mathbf{e}} \\
\mathbf{p}_d^{(+)}(t,\delta t/2) &= \left(1-\frac{i\delta t}{2(\Delta x)^2}\right)\hat{\mathbf{e}} - \frac{i\delta t}{2}\mathbf{V}(t+\delta t/2) \\
\mathbf{p}_u^{(+)}(t,\delta t/2) &= \frac{i\delta t}{4(\Delta x)^2}\hat{\mathbf{e}}
\end{aligned} \tag{4.1}$$

and

$$\begin{aligned}
\mathbf{p}_l^{(-)}(t,\delta t/2) &= -\frac{i\delta t}{4(\Delta x)^2}\hat{\mathbf{e}} \\
\mathbf{p}_d^{(-)}(t,\delta t/2) &= \left(1+\frac{i\delta t}{2(\Delta x)^2}\right)\hat{\mathbf{e}} + \frac{i\delta t}{2}\mathbf{V}(t+\delta t/2) \\
\mathbf{p}_u^{(-)}(t,\delta t/2) &= -\frac{i\delta t}{4(\Delta x)^2}\hat{\mathbf{e}}
\end{aligned} \tag{4.2}$$

The propagated wave function vector is then found from the system

$$\mathbf{P}^{(-)}(t,\delta t/2)\mathbf{y}(t+\delta t) = \mathbf{r} \tag{4.3}$$

by calling **Ctridag,** a complex version of the Numerical Recipes tridiagonal solver **tridag** [1], with the rhs vector given by

$$\mathbf{r} = \mathbf{P}^{(+)}(t,\delta t/2)\mathbf{y}(t) \tag{4.4}$$

The potential at point x and time t is supplied by a function **pot**.

3.2 Function **Propagate**

For the reasons outlined in section 1.1, the basic CCN algorithm implemented in function **Prop_CCN** needs a driver. This driver is function **Propagate**.

This **Propagate** function takes the wave function vector $\mathbf{y_1} \equiv \mathbf{y}(t_1)$ at time t_1 and evaluates the wave function vector $\mathbf{y}_2 \equiv \mathbf{y}(t_2)$ at time $t_2 > t_1$ by calling the CCN propagator function **Prop_CCN**. However, the time difference $\Delta t = t_2 - t_1$ may be too large for the CCN algorithm to be used directly with sufficient accuracy. In fact (see [1], [2]), the increment δt over which the CCN algorithm can accurately used is related to the x-mesh spacing Δx. We use an initial estimate $\delta t = a(\Delta x)^2$ for this increment, where the factor a is of order 1. We can then estimate the number of intermediate time steps needed to propagate accurately between t_1 and t_2. Function **Prop_CCN** is then called to propagate from t_1 to $t_1 + \delta t$, then from $t_1 + \delta t$ to $t_1 + 2\delta t$, and so on until time t_2 is reached.

How accurate is this procedure? We can form an error estimate by repeating the propagation with twice as many intermediate steps (ie by halving the factor a and thus the increment δt) and evaluating the rms deviation between the two results. This doubling of the number of intermediate time steps is continued until the rms deviation between the wave function vectors is less than a preset accuracy (set to 10^-6 in this version).

4 Tests

This section gives details of tests of the **TDSE1** code against previous completely independent calculations using the Wave-Packet Dynamics (WPD) code [5], which uses variants of the energy eigenstate method [6]. This method applies when the Hamiltonian does not depend on time. The idea is to write the wave function as a linear combination, ie a wave-packet, of energy eigenstates:

$$\psi(x,t)=\sum_i a_i\varphi_i(x)e^{-i\varepsilon_i t} \tag{6.1}$$

Here the solutions of the time-independent Schrödinger equation are

$$\mathcal{H}|\varphi_i\rangle=\varepsilon_i|\varphi_i\rangle$$
$$\varphi_i(x)=\langle x|\varphi_i\rangle$$
$$\langle x|\mathcal{H}|x'\rangle=H(x)\delta(x-x')$$

The above assumes a discrete spectrum; for continuous spectra, sums become integrals in the usual way. The wave-packet (6.1) automatically satisfies the TDSE

$$i\frac{\partial}{\partial t}\psi(x,t)=H\psi(x,t)$$

So the wave-packet dynamics algorithm, so to speak, is to find the eigenstates of *H* and combine them using the quadrature (6.1), subject to a given initial state $\varphi(x,t=0)$, which then gives the evolution for $t>0$.

4.1 Free plane-wave-packets

In this section we compare **TDSE1** calculations with those using the Wave-Packet Dynamics code for **FREE** space – ie no potentials, fields etc – "FREE" because this is the mode in which the WPD code was operated. The latter code evaluates the wave-packets dynamics (WPD) equation in $\mathbb{R}^1$

$$\psi(x,t)=(2\pi)^{-1/2}\int dk\, a(k)e^{ikx}e^{-ik^2t/2} \tag{6.2}$$

for stationary or moving wave-packets. Hence this is a test of the way the TDSE1 code handles free motion.

We choose the envelope function to be a Gaussian of width *l* , peaked at $k=k_0$ and scaled so that $\psi(x,t=0)$ is normalised to unity:

$$a(k)=\left(\frac{l}{\sqrt{\pi}}\right)^{1/2}e^{-(k-k_0)^2l^2/2} \tag{6.3}$$

which leads to

$$\psi(x,0)=\left(l\sqrt{\pi}\right)^{-1/2}e^{-x^2/2l^2}e^{ik_0x} \tag{6.4}$$

to be compared to (3.1).

4.1.1 Short-time behaviour

It is of interest to see how a stationary wave-packet at $t=0$ evolves for short times because we can do a short-time expansion analytically. Expand the WPD equation in powers of t :

$$\begin{aligned}\psi(x,t)&=\int dk\,a(k)e^{ikx}e^{-ik^2t/2}\\&=\int dk\,a(k)e^{ikx}\left(1-it\frac{k^2}{2}+O\left(t^2\right)\right)\\&=\psi(x,0)-it\psi^{(1)}(x)+O(t^2)\end{aligned} \tag{6.5}$$

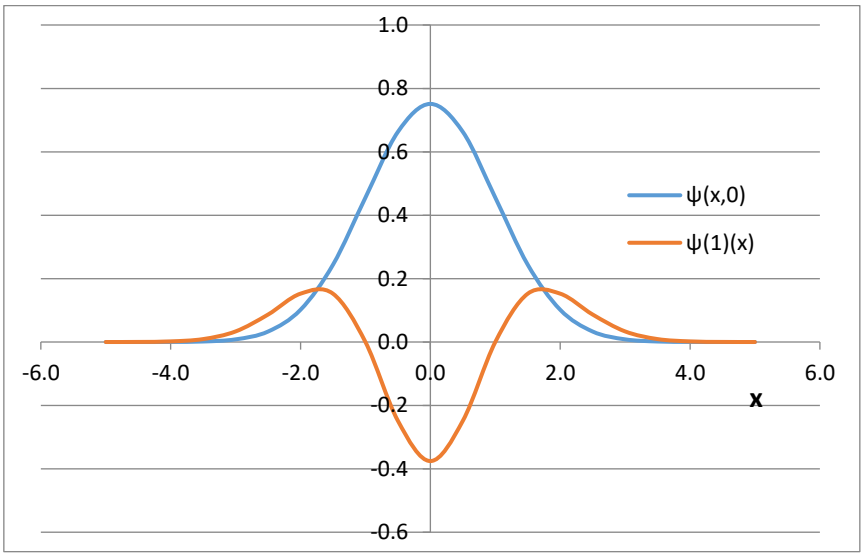
where

$$\psi^{(1)}(x)=\int dk\,a(k)\frac{k^2}{2}e^{ikx} \tag{6.6}$$

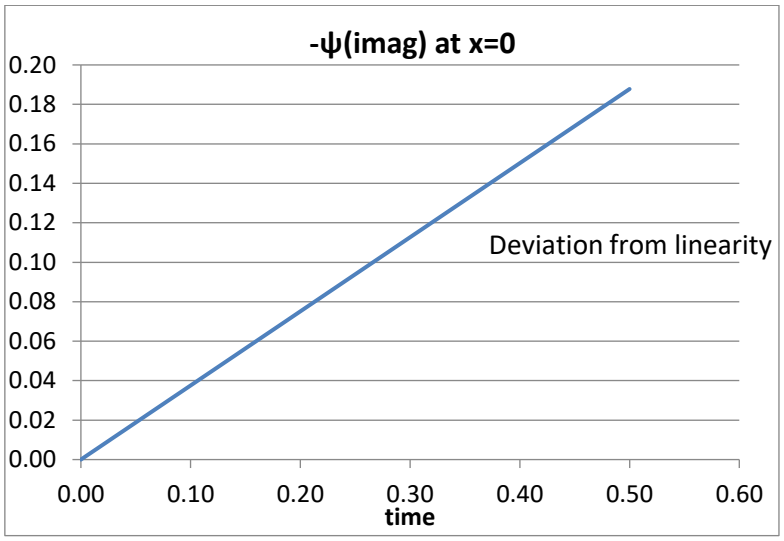
Let's take the envelope function to be the normalised Gaussian defined above in (6.3) with $k_0=0$. Then, by the standard tricks of Gaussian integrals (ie differentiating the integral with respect to l - see **Error! Reference source not found.**) it's easy to show that

$$\psi^{(1)}(x)=\frac{1}{2l^5}\left(\frac{l}{\pi^{1/2}}\right)^{1/2}\left(x^2-l^2\right)e^{-x^2/2l^2} \tag{6.7}$$

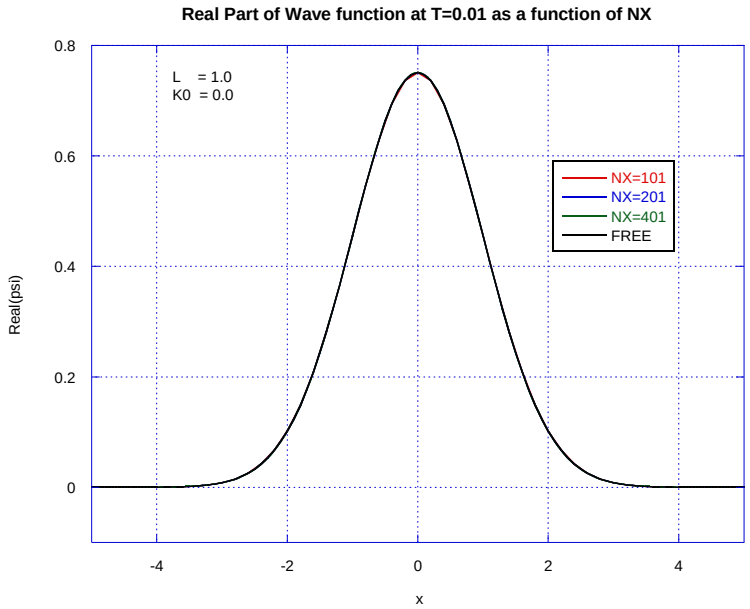
These functions look like this

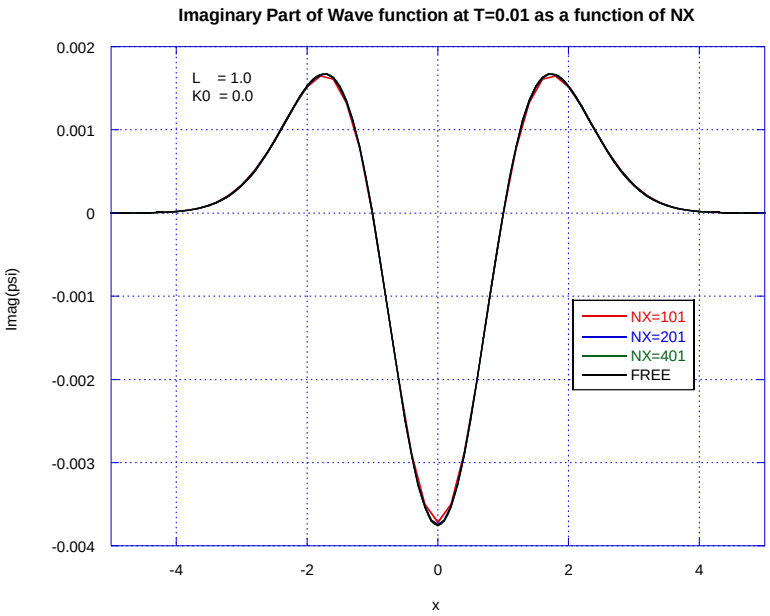


Full WPD calculations give exactly this behaviour. When does the small-time expansion break down? The following diagram shows the difference between a full WPD calculation and the linear approximation as a function of time for $x = 0$:

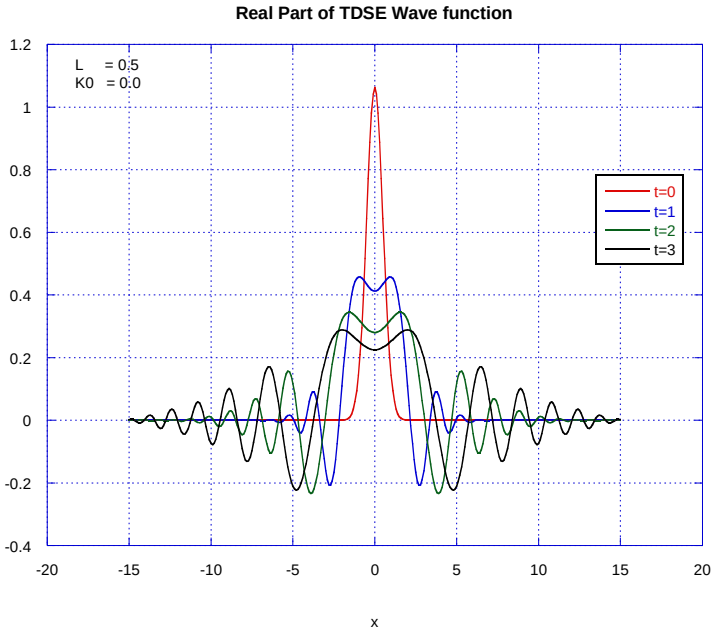


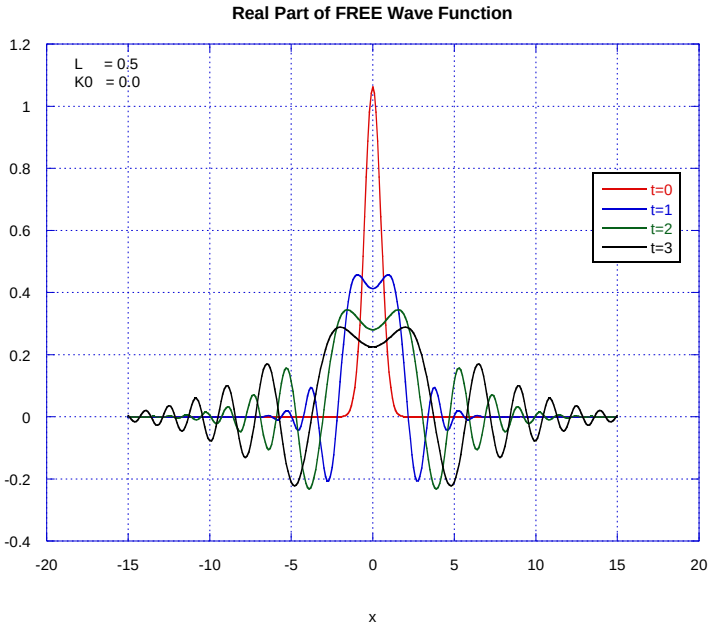
The TDSE1 code also gives this behaviour, as the following results for $t = 0.01$ show. Note that according to the above figure, the linear approximation should be OK for this time, and so TDSE1 should be even better, being based on a second order algorithm – see section 2.1. Note also that these calculations show the convergence with the number of x-mesh points N_x.

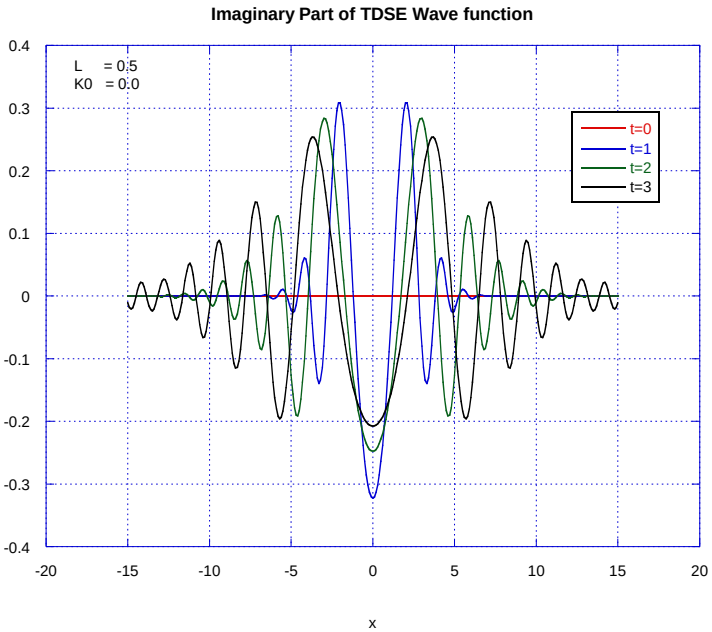


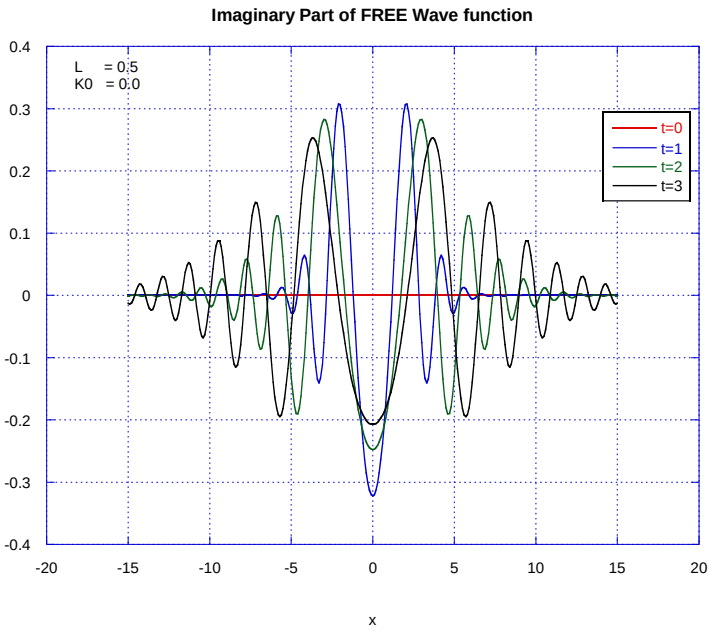


The following graphs show the time-evolution of a stationary wave-packet ($k_0 = 0$) as determined by the TDSE1 and WPD methods.

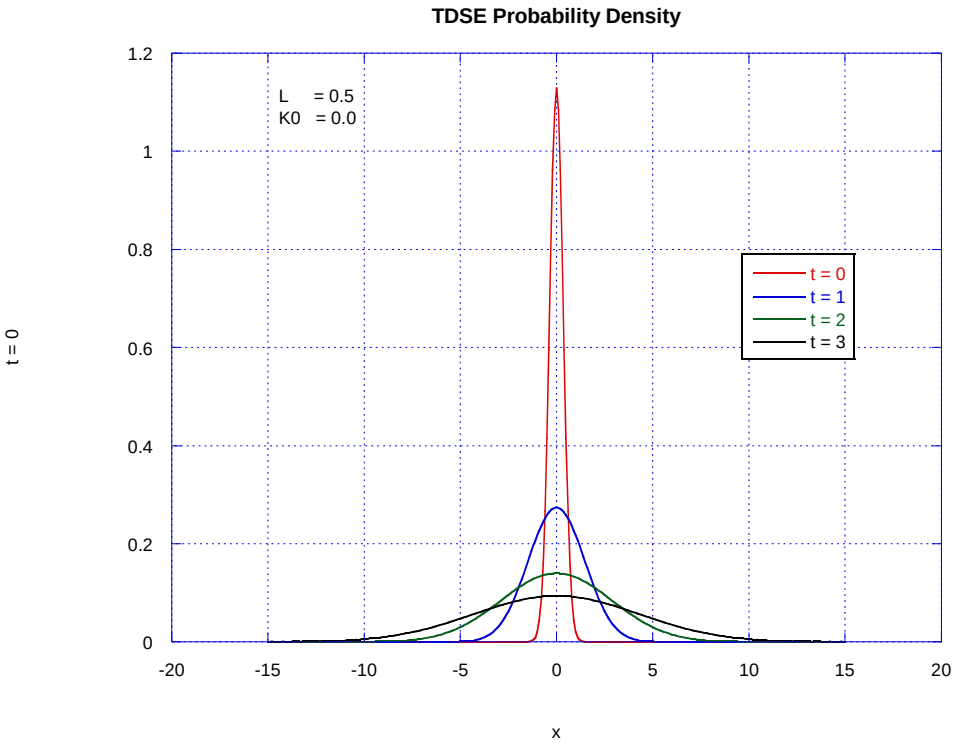


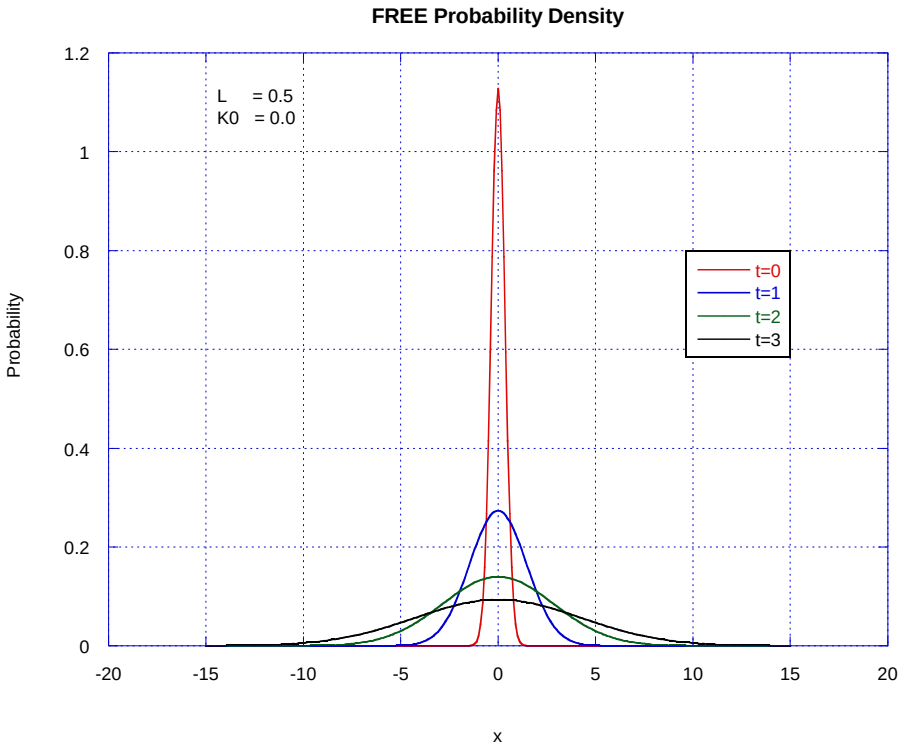

Real Part of FREE Wave Function
L = 0.5
K0 = 0.0
t=0
t=1
t=2
t=3
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-20
-15
-10
-5
0
5
10
15
20
x


Imaginary Part of TDSE Wave function
L = 0.5
K0 = 0.0
t=0
t=1
t=2
t=3
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-20
-15
-10
-5
0
5
10
15
20
x



The following two graphs show the time-evolution of the density.





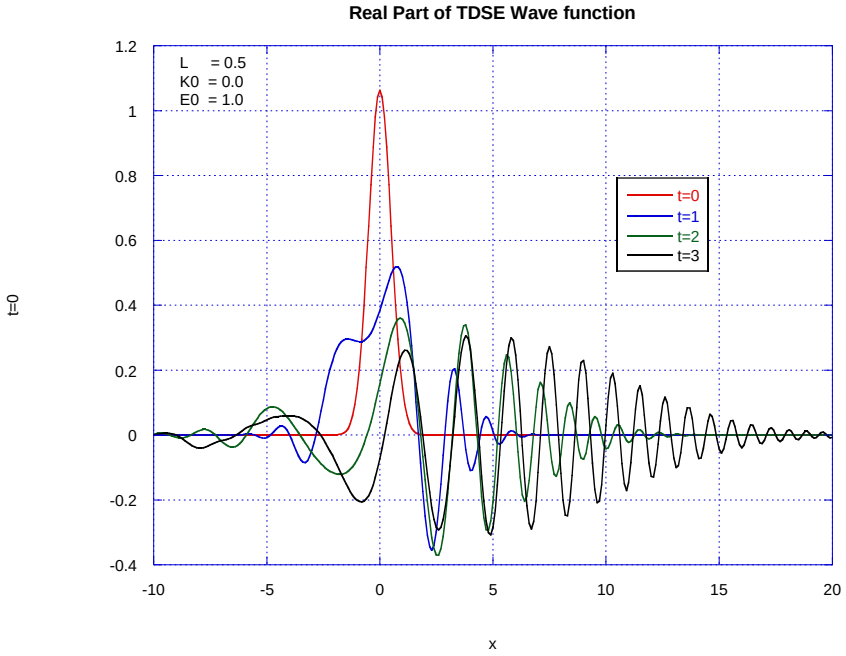
Clearly, all these calculations lie on top of each other. My focus here is testing the TDSE1 code, but I'll make a quick comment: the physics illustrated by the last two figures is the well-known time evolution of a free wave-packet with zero velocity [6]; it stays where it is but broadens (ie wave-packet spreading).

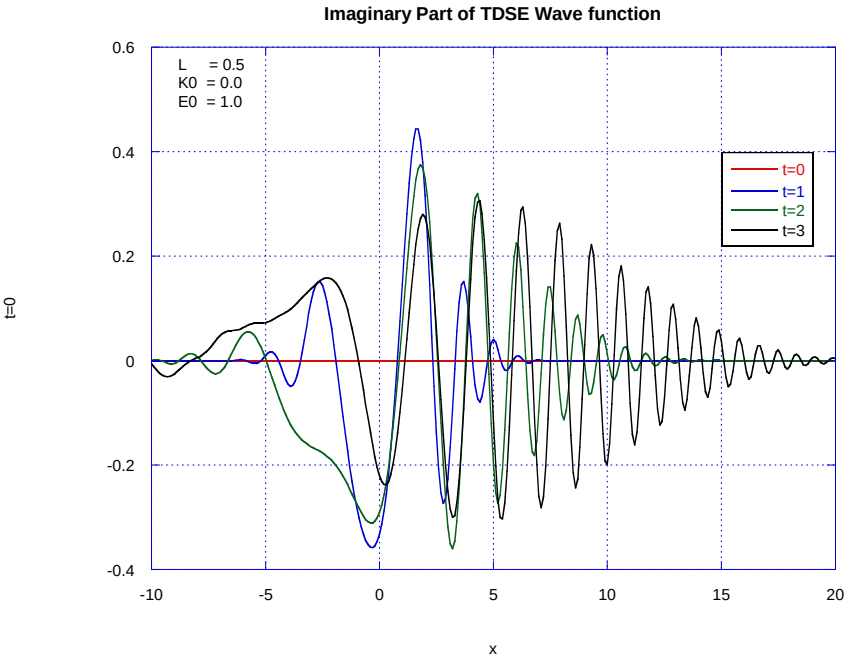
4.2 Static uniform field dynamics

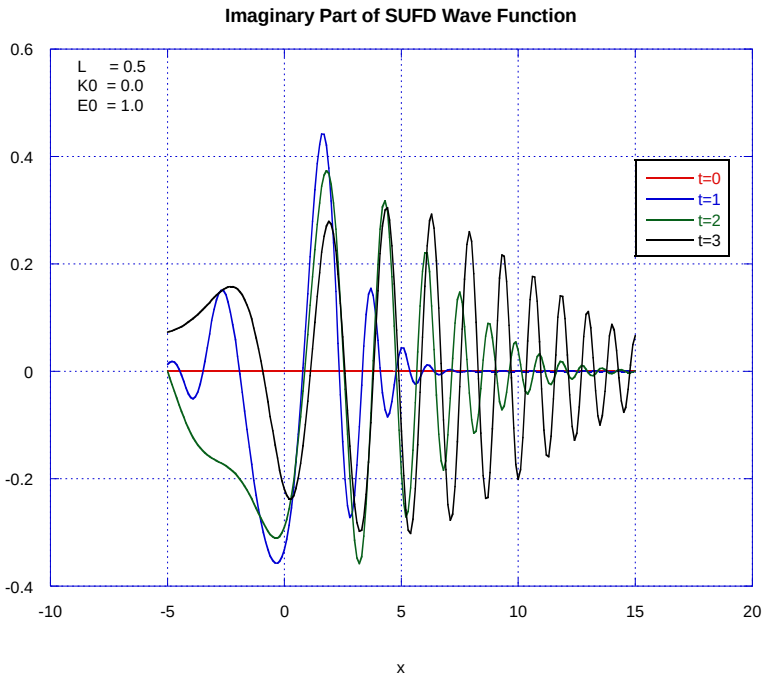
The graphs in this section make the same comparison for wave-packets accelerated by a static uniform field between the results of TDSE1 and those of the WPD code operating in **SUFD**[2] mode [5], [7]. Since $k_0 = 0$ here, we are accelerating the wave-packet from a standing start.

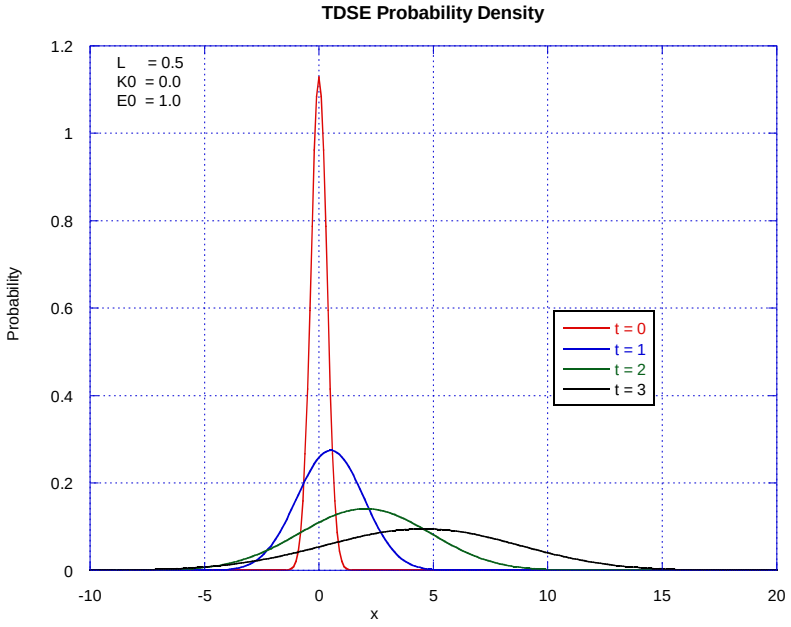
Again, the two sets of results fall on top of one another.

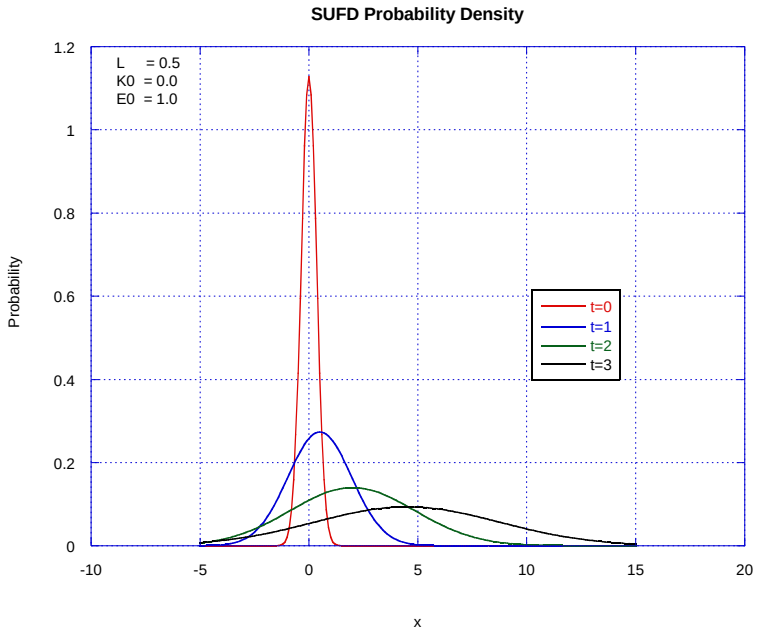
[2] A static uniform field corresponds to a linear potential. The spectrum is continuous and the corresponding energy eigenfunctions needed for (6.1) are Airy functions.


Real Part of TDSE Wave function
L = 0.5
K0 = 0.0
E0 = 1.0
t=0
t=1
t=2
t=3
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
15
20
t=0
x


Imaginary Part of TDSE Wave function
L = 0.5
K0 = 0.0
E0 = 1.0
t=0
t=1
t=2
t=3
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
15
20
t=0
x

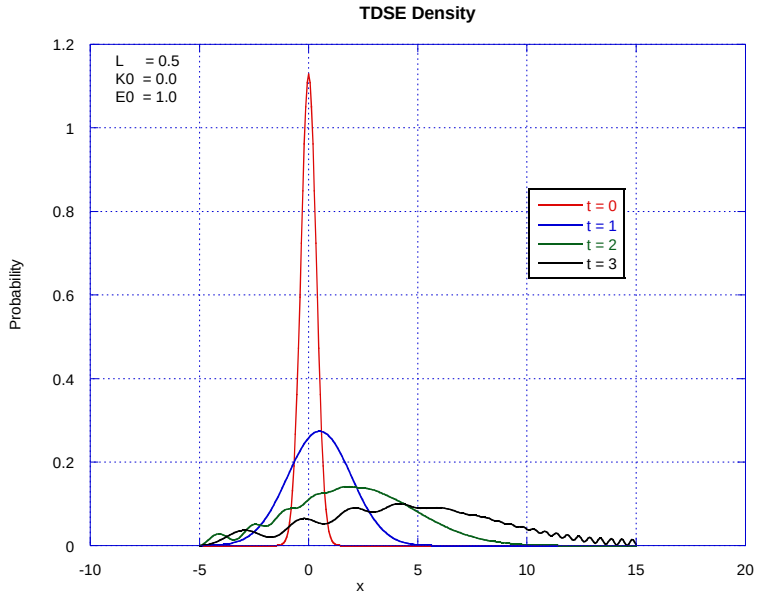

Imaginary Part of SUFD Wave Function
L = 0.5
K0 = 0.0
E0 = 1.0
t=0
t=1
t=2
t=3
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
15
20
x


TDSE Probability Density
L = 0.5
K0 = 0.0
E0 = 1.0
t = 0
t = 1
t = 2
t = 3
Probability
1.2
1
0.8
0.6
0.4
0.2
0
-10
-5
0
5
10
15
20
x



So, again, everything is fine. The physics in the last two figures is the acceleration (and spreading) of the wave-packet to the right.

But here's an interesting thing that's worthwhile noting.



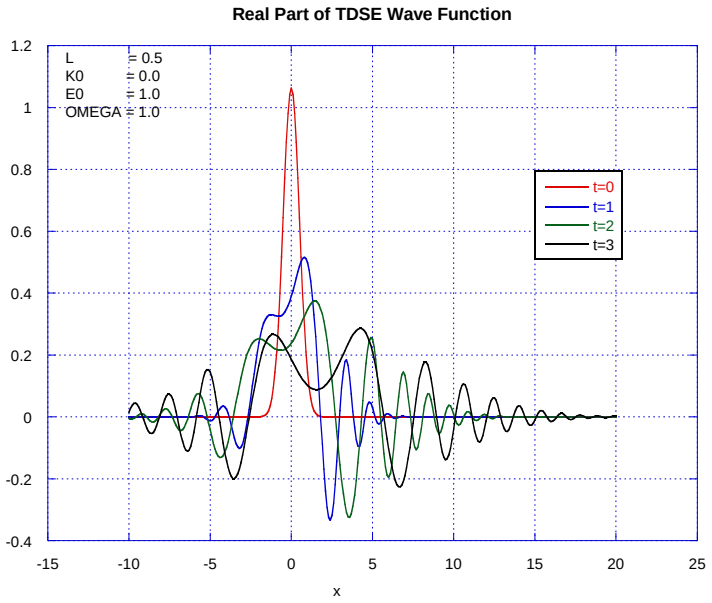
In these **TDSE1** calculations for the static uniform field problem, I made the calculations on an x-mesh that ran over the interval $-10 \le x \le 20$. The wave function is very small at the end points of this mesh and the results agree well with those of the WPD code in the **SUFD** mode. However I originally used an x-mesh over the interval $-5 \le x \le 15$. As the graphs above show, the wave functions are not at all small at the end points of this mesh. This should be a problem, because my implementation of the CCN differencing scheme assumes that the wave function vectors are zero at the end points of the mesh. And indeed it is – the figure above shows what happens to the density when calculated on this too small mesh:

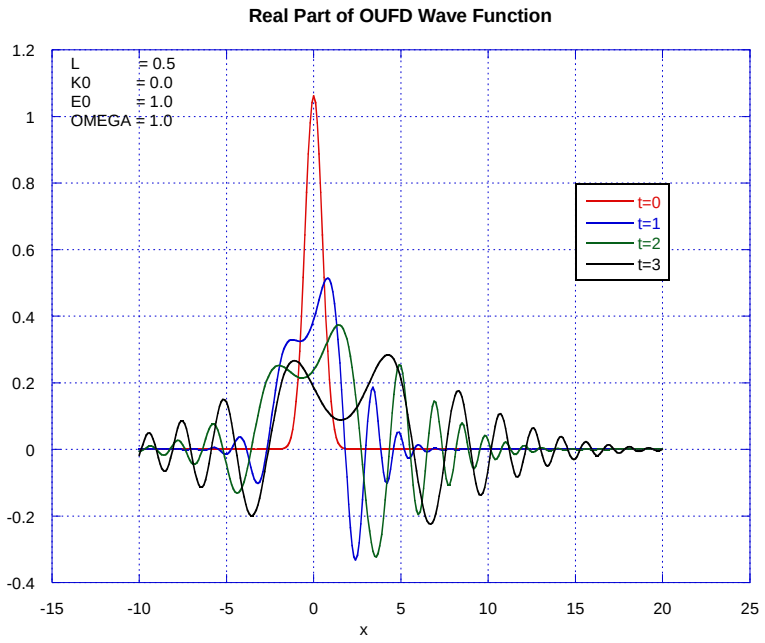
Maybe this disease could be cured by some kind of absorbing boundary condition, but for my purposes it seems safer simply to use a mesh covering a big enough region to enclose the whole wave function.

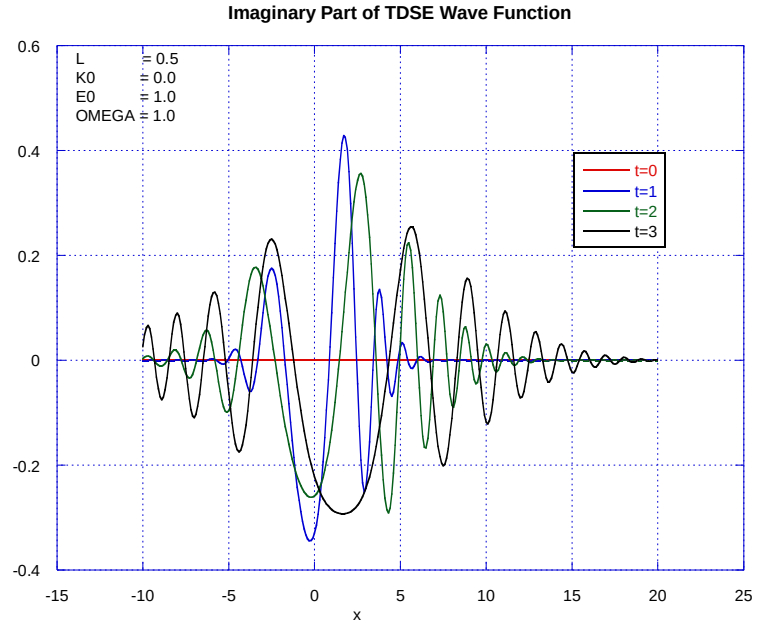
4.3 Oscillatory uniform field dynamics

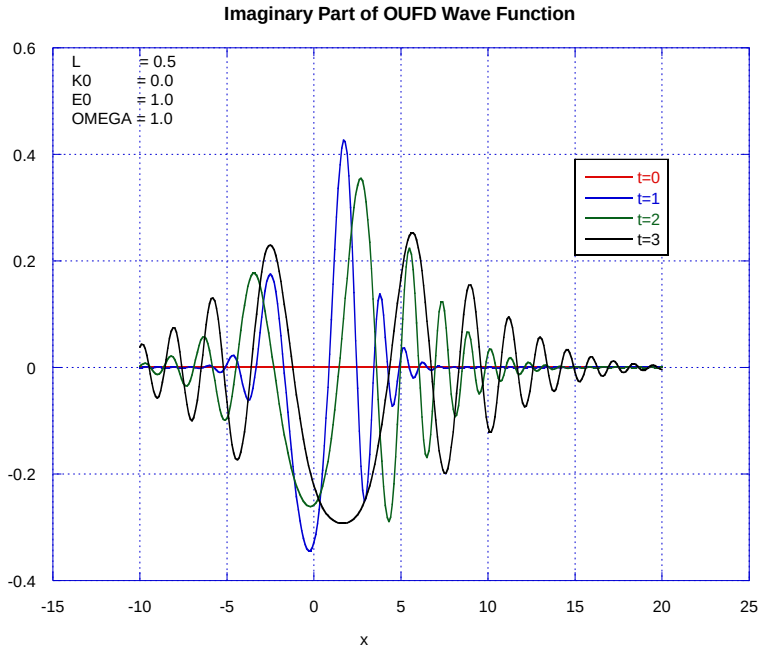
The graphs in this section make the same comparison for wave-packets accelerated by a oscillating uniform field between the results of **TDSE1** and those of the WPD code operating in **OUFD** mode [5], [7]. This toy model bears some similarity to the problem of the ionisation from a bound state induced by a strong laser field.

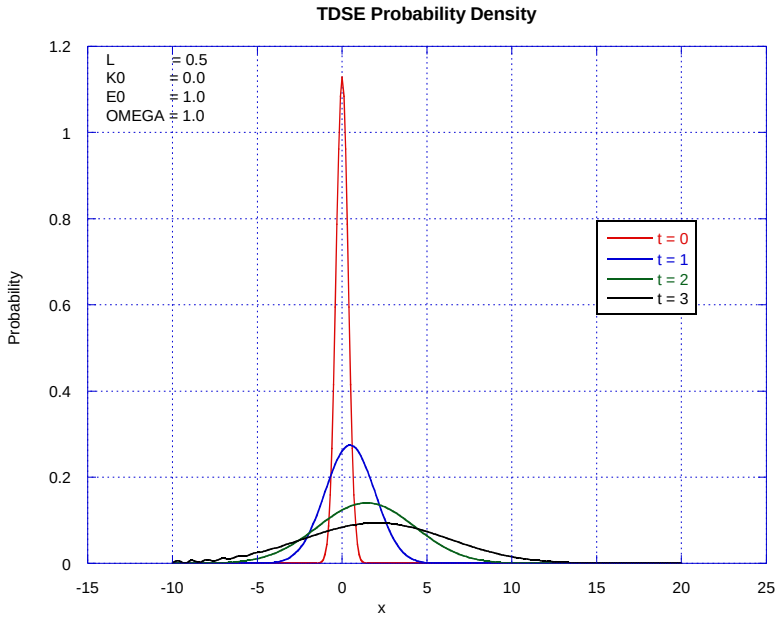
This is the first test case which involves a time-dependent Hamiltonian, and again the results are in excellent agreement. Note that there is a gauge choice to be made in this and the previous (ie **SUFD**) case – see [7]. Since the **TDSE1** code works with a time-dependent scalar potential, the wave functions it produces must be compared with WPD results in a gauge in which the vector potential is zero. In the WPD code, however, it's much easier to solve the problem in a gauge in which the scalar potential is zero. The quantum transformation between gauges is straightforward. This is quite an interesting point that is explained in my notes on accelerating wave-packets [7]; at some point I will probably turn those notes into another post in this series.

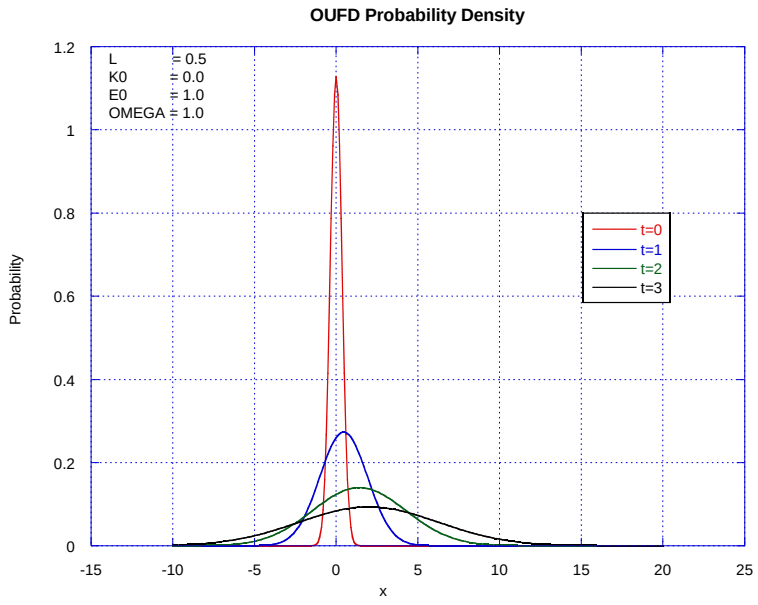



Real Part of OUFD Wave Function
L = 0.5
K0 = 0.0
E0 = 1.0
OMEGA = 1.0
t=0
t=1
t=2
t=3
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15
20
25
x


Imaginary Part of TDSE Wave Function
L = 0.5
K0 = 0.0
E0 = 1.0
OMEGA = 1.0
t=0
t=1
t=2
t=3
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15
20
25
x


Imaginary Part of OUFD Wave Function
L = 0.5
K0 = 0.0
E0 = 1.0
OMEGA = 1.0
t=0
t=1
t=2
t=3
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15
20
25
x


TDSE Probability Density
L = 0.5
K0 = 0.0
E0 = 1.0
OMEGA = 1.0
t = 0
t = 1
t = 2
t = 3
Probability
1.2
1
0.8
0.6
0.4
0.2
0
-15
-10
-5
0
5
10
15
20
25
x



Here the physics is similar to that of the SUFD calculations of the last section, but now the initial acceleration of the wave-packet to the right slows down as the field changes direction. After more time elapses it will start to move to the left [7].

5 Imaginary time propagation – the **ITP** code

Here's another interesting thing that can be done with the time-dependent Schrödinger equation: find bound states (see, for example, [8]).

First let's recap what we know about wave-packets. Suppose we have a time-independent Hamiltonian H with eigenstates and eigenvalues $\varphi_i(x)$, ε_i:

$$H\varphi_i(x) = \varepsilon_i\varphi_i(x), \quad i = 0,1,2,3,.... \tag{7.1}$$

Let's take any linear combination of these eigenstates as the initial $t = 0$ wave function:

$$\psi(x,t=0) = \psi(x,0) = \sum_{i=0} a_i\varphi_i(x) \tag{7.2}$$

Now since the Hamiltonian is static, the solution of the TDSE in the form of the evolution operator is just $U(t) = e^{-iHt}$, such that

$$\begin{aligned}\psi(x,t) &= U(t)\psi(x,0)\\ &= \sum_{i=0} a_i\varphi_i(x)e^{-i\varepsilon_i t}\end{aligned} \tag{7.3}$$

Suppose that we propagate the wave function through a sequence of imaginary times:

$$t_n = -iT_n = -in\tau, \quad n = 1,2,3.... \tag{7.4}$$

where τ, T_n are real and positive. In other words we march down the negative imaginary time axis in steps of length τ. Then, from (7.3), we can write

$$\psi_n(x) \equiv \psi(x,t_n) = \psi(x,-in\tau) = \sum_{i=0} a_i \varphi_i(x) e^{-n\varepsilon_i \tau} \tag{7.5}$$

Now suppose that the spectrum of *H* has a discrete part such that the state $\varphi_0(x)$ is the lowest lying bound state. Then

$$\psi_n(x) = e^{-n\varepsilon_0\tau}(a_0\varphi_0(x) + \sum_{i=1} a_i \varphi_i(x) e^{-n\Delta\varepsilon_i\tau}) \tag{7.6}$$

where $\Delta\varepsilon_i = \varepsilon_i - \varepsilon_0 > 0$. Thus as *n* increases, the contributions from all states other than the lowest bound state die out, and we have

$$\lim_{n\to\infty} \psi_n(x) = a_0\varphi_0(x) e^{-n\varepsilon_0\tau} \tag{7.7}$$

Now the point of this is that if one solves the TDSE *numerically* for the sequence of imaginary times (7.4) starting from ***any*** initial wave function (see (7.2)), we will generate a sequence of wave functions that converges to that of the lowest bound state.

From (7.7), we can find $\varphi_0(x)$ simply by normalising $\lim_{n\to\infty} \psi_n(x)$. A simple way to find the eigenvalue is to look at the norms of the sequence of wave functions $\psi_n(x)$. Define

$$N_n = \int dx \left|\psi_n(x)\right|^2 \tag{7.8}$$

Then, since $\varphi_0(x)$ is normalised to unity by hypothesis, we easily see from (7.7) that

$$\lim_{n\to\infty} N_n = \left|a_0\right|^2 e^{-2n\varepsilon_0\tau} \tag{7.9}$$

and thus

$$\lim_{n\to\infty} \frac{N_{n-1}}{N_n} = e^{2\varepsilon_0\tau} \tag{7.10}$$

from which ε_0 is easily found.

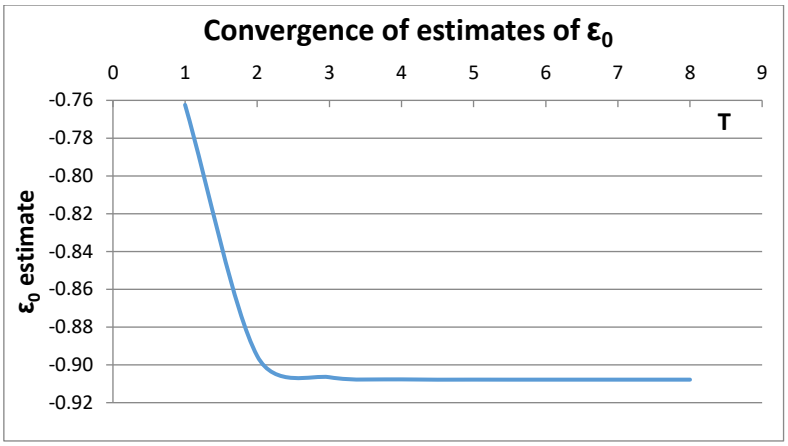
Finally, if one wishes to find the other bound states, having found the lowest, one can simply propagate again through the sequence of imaginary times starting with an initial wave function that is orthogonal to $\varphi_0(x)$. Thus, in terms of (7.2), we can define

$$\begin{aligned} \psi'(x,0) &= \psi(x,0) - \varphi_0(x)\int dx\, \varphi_0^*(x)\psi(x,0) \\ &= \sum_{i=1} a_i \varphi_i(x) \end{aligned} \tag{7.11}$$

Since $\psi'(x,0)$ contains no contribution from the lowest bound state, the imaginary time propagation procedure described above will project out the lowest remaining bound state, ie $\varphi_1(x)$. And so on for the other bound states.

Finally, what physical significance can we attribute to this imaginary time propagation? One way to think about this is to compare imaginary time to temperature, recalling the analogy between the evolution operator and the Boltzmann distribution[3]. The initial wave function is a bit like an arbitrary distribution over the eigenstates of the Hamiltonian, and the imaginary time propagation then represents a process of cooling down so that the system ends up in its lowest energy state. But this is pretty loose talk. For one thing, the initial "distribution" is not necessarily thermal (ie an equilibrium distribution corresponding to some temperature) – it can be anything. For another, although the connection between the imaginary time evolution operator and the Boltzmann factor is easy to see, what is the "thermal" equivalent of the TDSE itself? What's the corresponding PDE, with temperature as an independent variable, in statistical physics? Diffusion? All these questions signal that it's a bit too much to regard imaginary time propagation as equivalent to varying the temperature of an equilibrium system in any real physical sense. But it's still an interesting analogy.

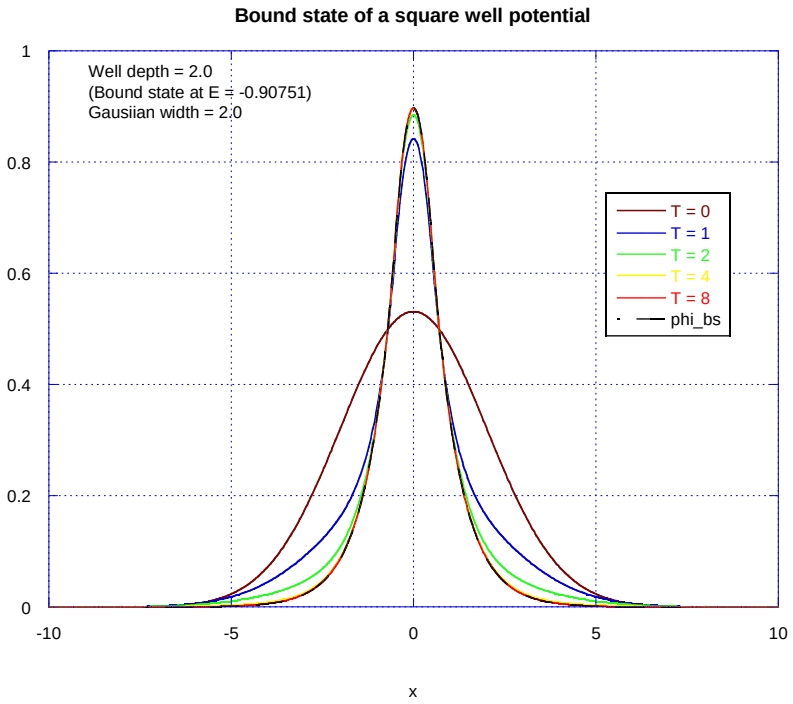
I implemented this procedure in the **ITP** code, which is a very straightforward modification of the **TDSE1** code. Here's an example for a [1-D] square well potential of depth 2 and width 1 in atomic units – this potential has a single bound state at energy -0.90751. In the following calculations the initial state was taken to be a normalised Gaussian of width 2 and the time step down the imaginary time axis was $\tau = 1$. First, here's how the eigenvalue estimates converge through the timesteps.



It should be noted that in these calculations the converged bound state eigenvalue was -0.90781 – not exactly what it should be. I have not yet really looked at the numerical accuracy of the **ITP** code but at least it does work.

Next, the sequence of wave functions is compared with the "true" bound state wave function (called "phi_bs") as determined by a conventional wave function matching calculation (as, of course, is the "true" bound state energy). The convergence to the correct solution is clear.

[3] Of course, there are proper ways of formally establishing the relationship between imaginary time and temperature [9]. Here I'm using it somewhat imaginatively.


Bound state of a square well potential
Well depth = 2.0
(Bound state at E = -0.90751)
Gausiian width = 2.0
T = 0
T = 1
T = 2
T = 4
T = 8
phi_bs
1
0.8
0.6
0.4
0.2
0
-10
-5
0
5
10
x

References

[1] “Numerical Recipes in C++ (second edition)”, W H Press, S A Teukolsky, W T Vetterling and B P Flannery (CUP 2003)
[2] Numerical solution of PDEs v1.0.docx, PJD (DL 2010)
[3] Elementary Quantum Dynamics, PJD (DL 2019)
[4] “Modern Quantum Mechanics”, J J Sakurai (Addison-Wesley, 1994)
[5] Wave-Packet Dynamics Code Notes.docx, PJD (DL 2010)
[6] "On-wave-packet-dynamics", CoSeC Blog, Paul Durham (DL 2020)
[7] Accelerating Wave-Packets v2.0.docx, PJD (DL 2010)
[8] R Kosloff and H Tal-Ezer, Chem Phys Letters, **127**, 223 (1986)
[9] “Quantum Statistical Mechanics”, L P Kadanoff and G Baym (CRC Press, 2018)