SHPC4001 - Assignment 6

Ayden S. McCann

April 2021

Code Listing at the end

Introduction

The ability to numerically calculate the time propagation of quantum systems is an important tool which differs from analytic calculations in various ways. As in earlier works in this unit which detailed the forwards, backwards and central difference methods for numerical differentiation, there are methods of propagating the Schrödinger equation forwards in time which utilise the same train of thought. However, the works of this assignment focus on the 4th order Runge-Kutta method detailed in Session 6. In order to demonstrate the viability of this method a wave packet (2) was propagated forwards in time and its interaction with a potential barrier (1) analysed.

1 Question 1

1.1

a Using the code in 1.1a.py, the time-independent potential barrier (1) was plotted over the domain $0 \leq x \leq 100$. This is shown below in Figure 1.

$$V(x) = \frac{V_0}{cosh^2(\frac{x-x_0}{\omega})} \tag{1}$$

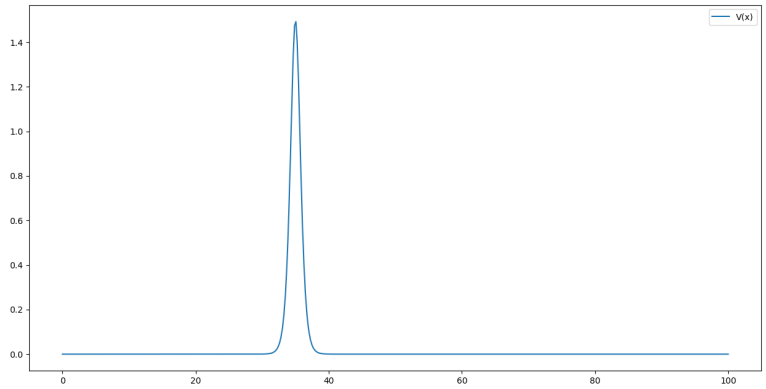

Figure 1: The potential (1) plotted over the domain $0 \leq x \leq 100$.

b Using this same potential, the dynamics of a Gaussian wave packet incident on the potential barrier (from the left), were simulated. The code 1.1bc.py was based initially off of the code supplied as Example 11.2 in [1]. The code in [1] utilised the fourth order Runge-Kutta method as desired for the purposes of this assignment. However, it featured a different $\psi(x,0)$ with no initial momentum, in a region of zero potential. This code was modified to incorporate the more complicated wave packet equation (2), and the potential plotted in Figure 1 was also introduced.

$$\psi(x,0) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{4}}} e^{-(x-x_0)^2/4\sigma^2} e^{ik_0 x} \tag{2}$$

Another important feature in 1.1bc.py is the addition of a Nyquist sampling theorem check. This theorem ensures that the momentum components can be completely determined, so long as the condition on (3) is upheld. If this condition is violated through the selection of any relevant parameters an exception is raised informing the user of this violation, before stating the relevant maximum value of dx that would be permissible with the given choices of k_0, V_{max}, σ and ϵ.

$$\Delta x \leq \left[k_0 + \sqrt{2V_{max}}/\hbar + \frac{1}{\sqrt{2}\sigma} \sqrt{ln\left(\frac{\sigma}{\epsilon}\sqrt{\frac{2}{\pi}}\right)} \right]^{-1} \tag{3}$$

c After this simulation is performed, the remainder of 1.1bc.py is dedicated to creating an animation, so that the interaction between the wave packet and the potential barrier can be seen developing through

time, this was performed using the matplotlib.animation package. The animation is controlled via the following introduced parameters: Scale allows for the arbitrary vertical scaling of the wave packet, to allow for meaningful superposition with the potential; Frames dictates the number of frames in the animation; fps dictates the frames per second of the animation and finally speed allows frames to be skipped in the production of the animation. This was introduced to avoid raising dt and risking consequent instabilities. Instead of the animation plotting every ith result, it takes every $speed * i^{th}$ result. This way the system can be iterated over long time periods without needing to resort to ridiculously high frames and fps to reach the latter stages in a reasonable amount of time.

A .gif produced using speed = 100, frames = 50 is supplied on github. The speed parameter introduced allows frames to be skipped in the production of the animation while not changing raising dt and risking consequent instabilities. Instead of the animation taking every ith result, it takes every $speed * i^{th}$ result.

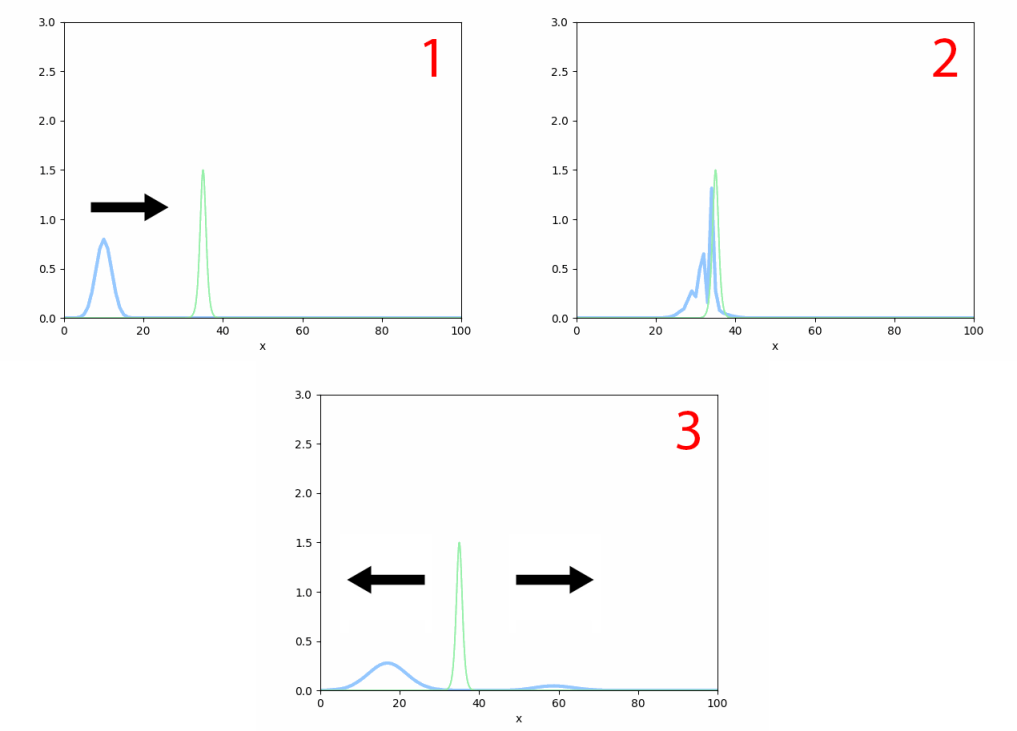

Figure 2: The probabilistic evolution of the wave packet $|\phi(x,t)|^2$ over the domain $0 \leq x \leq 100$ in the presence of the introduced potential (1).

The main stages in the evolution of the wave packet present in the supplied .gif have been summarised in Figure 2 into 3 key stages:

(1) The Gaussian wave packet has initial momentum in the positive x direction and is moving towards the potential barrier.

(2) The wave packet is incident on the potential barrier. As seen in Figure 2 this results in the compression of its probability density and a large peak.

(3) The result of the interaction between the wave packet and the potential barrier is that a large proportion of the incident wave packet probability density is reflected back in the negative x direction, while a smaller portion succeeds in overcoming the barrier and continuing on in the positive x direction.

Conclusion

As demonstrated in the above works the 4th order Runge-Kutta method was utilised to simulate the time evolution of a Gaussian wave packet over the domain $0 \leq x \leq 100$. Considerations were made to ensure that there was no significant overlap between the wave packet and the potential at $t = 0$, and that all boundary conditions were met. Iterations of the system with the wave packet starting partially outside the domain incurred sinusoidal oscillations superimposed on the wave packets probability density (see appendix 1). All of the above work was interpreted through the use of an animation, using the matplotlib.animation package.

2 Code Listing

Question 1.1a: 1.1a.py

Question 1.1b and 1.1c: 1.1bc.py and potential_barrier.gif

References

[1] Izaac, J Wang, J 2018, Computational Quantum Mechanics. Undergraduate Lecture Notes in Physics, Springer, Netherlands.

3 Appendix

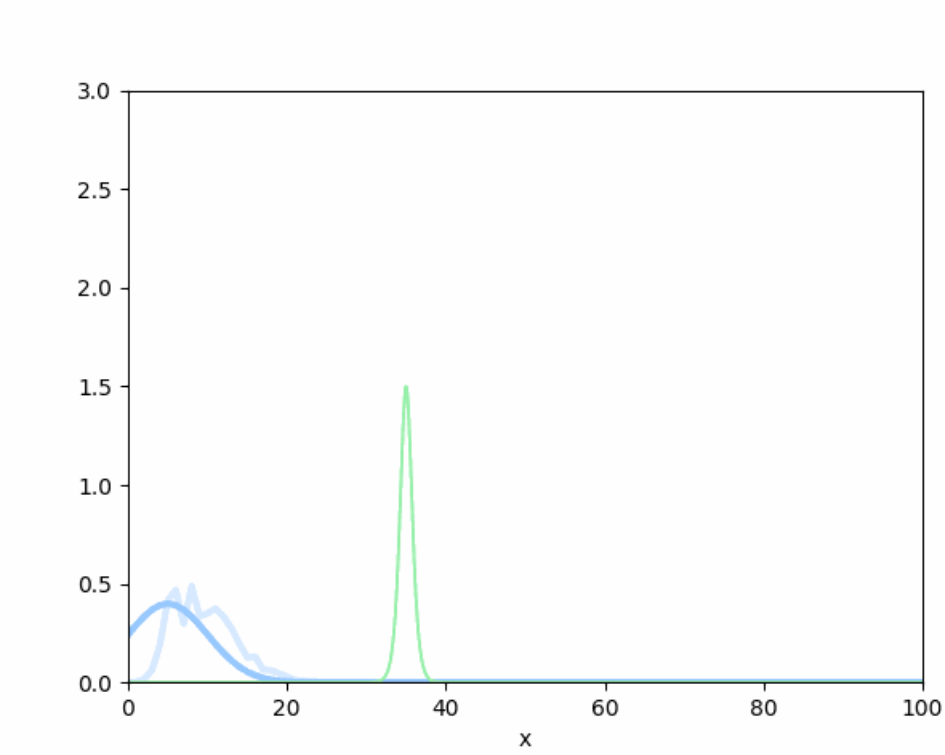

Figure 3: The result of the wavepacket at t=0 being severely truncated by the limits of the domain. Superimposed is the frame from a slightly later timestep, and the oscillatory motion described in the conclusion can be seen 'contaminating' the clean Gaussian shape evident in, for example, Figure 2.