PHYS3071 Assignment 2

Ryan White
s4499039

September 1, 2022

1 Question 1

As a numerical integration programming exercise, Simpson's rule was implemented in C++. Over some domain $x \in [a, b]$, Simpson's rule approximates the 'area under the curve' by

$$\int_a^b f(x)\ dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{1}$$

which effectively splits the domain into two and calculates the area under a quadratic approximation of the function in question. Simpson's rule heavily biases the midpoint, and so large step sizes ($h = b-a$) for functions that are not monotonic are problematic. To implement smaller step sizes of some domain, multiple iterations of Simpson's rule can be performed. For some domain $x \in [x_1,\ x_n]$,

$$\int_{x_1}^{x_n} f(x)\ dx = \int_{x_1}^{x_2} f(x)\ dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\ dx = \sum_{i=1}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\ dx \tag{2}$$

is how multiple steps is implemented.

In order to test how well this implementation worked, some test-cases were taken from quantum mechanics. To begin with, the solution to the one-dimensional Schrodinger equation reads as

$$\psi(x) = \mathcal{N}_n H_n(\sqrt{\alpha}x)e^{-\frac{\alpha x^2}{2}} \tag{3}$$

where $\mathcal{N}_n$ is the normalization constant for the Hermite polynomial H_n of degree n, and $\alpha = m\omega/\hbar$ (where m is the mass of the particle and ω is the angular frequency). Since the wave-function of the particle necessitates that it must be at *some* point in space, the integral of the 1D-Schrodinger equation solution (from negative infinity to infinity) must be equal to one, and hence the normalisation constant can be evaluated:

$$1 = \int_{-\infty}^{\infty} \psi(x)\ dx \tag{4}$$

$$= \int_{-\infty}^{\infty} \mathcal{N}_n H_n(\sqrt{\alpha}x)e^{-\frac{\alpha x^2}{2}} \tag{5}$$

$$\Rightarrow \mathcal{N}_n = \frac{1}{\int_{-\infty}^{\infty} H_n(\sqrt{\alpha}x)e^{-\frac{\alpha x^2}{2}}} \tag{6}$$

For the sake of simplicity and ease of calculation, the mass of the particle was set to $m = 9.11 \times 10^{-31}$ kg (rest mass of an electron) and the angular frequency $\omega = 1$. Hence, $\alpha = m_e/\hbar$. To assess the accuracy of the implementation, equation (6) was solved for the first four Hermite polynomials (i.e., order 0 up to 3 inclusive) over a range of bounds and stepsizes, and then compared against analytic solutions. The best and worse case scenarios are shown in Table 1

Order n	Accuracy	Steps	Integral Bounds	Normalisation Constant $\mathcal{N}_n$	Inverse Error (ϵ^{-1})
0	Best	10^6	$\pm 10^1$	37.0717	2.18724×10^{-8}
	Worst	10^3	$\pm 10^{-6}$	5×10^7	-0.0269727
1	Best	10^6	$\pm 10^1$	infinity	0
	Worst	10^6	$\pm 10^5$	5.54634×10^{10}	1.80299×10^{-11}
2	Best	10^6	$\pm 10^2$	18.5359	4.37449×10^{-8}
	Worst	10^3	$\pm 10^{-6}$	-2.5×10^5	-0.0539534
3	Best	10^6	$\pm 10^{10}$	infinity	0
	Worst	10^6	$\pm 10^5$	-9.2439×10^9	-1.08179×10^{-10}

Table 1: Results for Solving Equation (6)

Clearly, higher number of steps consistently yield better results. The worst results almost always come from small integral bounds, but also aren't great for unusually large bounds either. This is because the functions fall off as a Gaussian for sufficiently large Δx, and so small bounds don't approximate enough of the integral, while large bounds overshoot the integral.
As a further test case, the same numerical integration method was applied to a two-dimensional case: the paraxial wave equation whose solution is given by

$$\phi_\perp(x,y) = \mathcal{N}_n H_m\left(\frac{\sqrt{2}x}{\omega_0}\right) H_n\left(\frac{\sqrt{2}y}{\omega_0}\right) e^{-\frac{x^2+y^2}{\omega_0}} \tag{7}$$

As before, the normalisation constant can be found by approximating unity (and eventually separating the variables),

$$1 = \int_{-\infty}^{\infty} \phi_\perp(x,y) \tag{8}$$

$$= \iint_{-\infty}^{\infty} \mathcal{N}_n H_m\left(\frac{\sqrt{2}x}{\omega_0}\right) H_n\left(\frac{\sqrt{2}y}{\omega_0}\right) e^{-\frac{x^2+y^2}{\omega_0}}\, dx\, dy \tag{9}$$

$$= \mathcal{N}_n \iint_{-\infty}^{\infty} H_m\left(\frac{\sqrt{2}x}{\omega_0}\right) H_n\left(\frac{\sqrt{2}y}{\omega_0}\right) e^{-\frac{x^2+y^2}{\omega_0}}\, dx\, dy \tag{10}$$

$$\Rightarrow \mathcal{N}_n = \frac{1}{\iint H_m\left(\frac{\sqrt{2}x}{\omega_0}\right) H_n\left(\frac{\sqrt{2}y}{\omega_0}\right) e^{-\frac{x^2+y^2}{\omega_0}}} \tag{11}$$

$$= \frac{1}{\int H_m\left(\frac{\sqrt{2}x}{\omega_0}\right) e^{-x^2/\omega_0} \int H_n\left(\frac{\sqrt{2}y}{\omega_0}\right) e^{-y^2/\omega_0}} \tag{12}$$

Since the integrals multiply each other, we'd expect error inherently associated with each to be propagated. As such, the final error *could* be as prominent as the larger error squared, $\epsilon^2_{\max}$. In general, we'd expect the absolute error, $|\epsilon|$ to be as bad as $|\epsilon| \leq |\epsilon_{\max}|^n$, where n is the dimension of the integral problem.
Using bounds of $x \in [-10, 10]$ and 10^6 steps, each possible normalisation constant for any combination of Hermite polynomial (up to order 3 inclusive) was calculated (and is available after running the C++ submission code in the file `Q1b-Results.txt`).

2 Question 2

To generate a random walker on a discrete grid, a C++ function was first created which generated a pseudo-random number uniformly distributed on the interval $[0, 1)$. If the random variable was ‘0.25, the walker would decrease it's x position by 1 (or increase by 1 if $0.25 <$ r.v. ≤ 0.50). If the random variable was between 0.50 and 0.75, it would decrease it's y position by 1, and if greater than 0.75, it would increase it's y position by 1.

To test the randomness of the simulation, 400 walkers were generated and each took 50 steps. The resulting distribution of walkers at every time in their walk is shown in Figure 1.

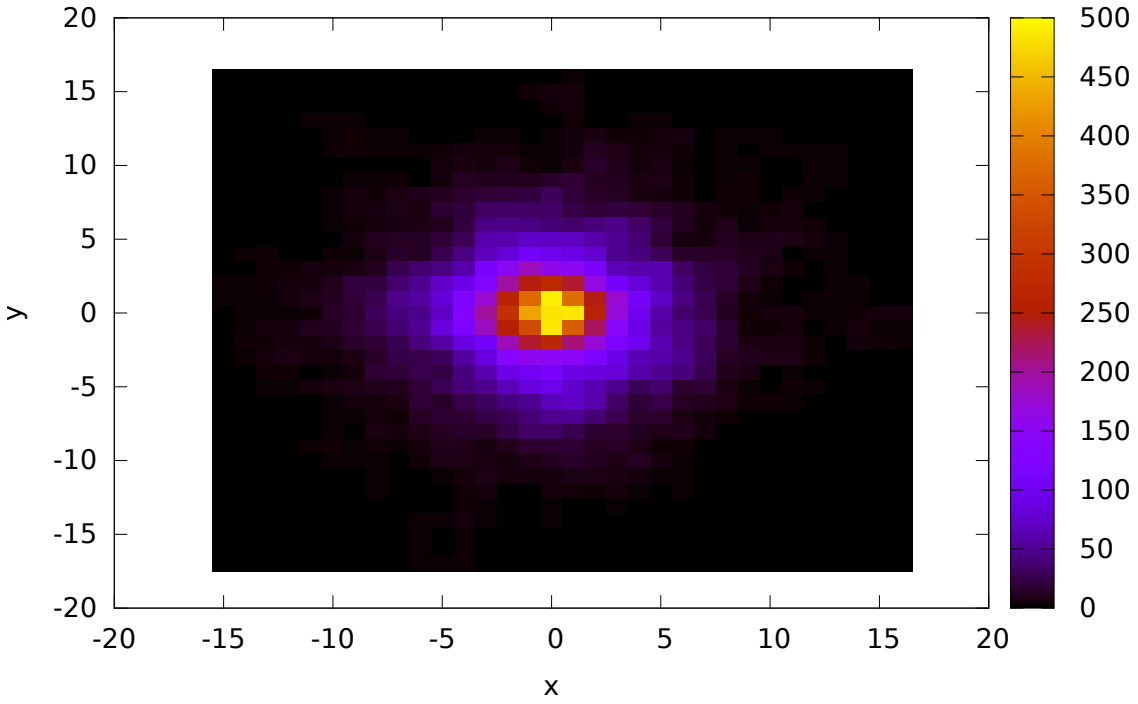

Figure 1: Random Walkers on a Discrete Grid

This approximates a two-dimensional Gaussian distribution (claim by eye), where the standard deviations were found to be $\sigma_x \simeq 3.47$ and $\sigma_y \simeq 3.56$, with a mean of $\mu_x = \mu_y = 0$. The mean square displacement of the walkers is calculated by

$$\text{MSD} \equiv \frac{1}{n}\sum_{i=1}^{n}(x(t) - x(0))^2 \tag{13}$$

where $t = 50$ (i.e. the final position of the walker) and $x(0) = (0, 0)$ (i.e. the starting position at the origin). For the simulation of 400 walkers with 50 steps, it was found that MSD $= 48.395$. When accounting for the walkers' displacement from the origin at the final step, the average speed (not accounting for direction) of the walkers was $\overline{v} \simeq 0.122795$ units / step. I claim (from observation of Figure 1) that if direction were to be taken into account, the average velocity across each axis would be approximately 0 (within some degree of error).

Using a very similar implementation as the example above, another, more complex, random walk simulation was performed on a discrete grid with predefined barriers. A large `.txt` file filled with a grid of 1001×1001 data points was supplied, where a value of 0 corresponded to a barrier and 1 an empty space. If a walker tried to walk into a barrier, their next step would be redrawn until they no longer wanted to walk into a wall. Using 40 walkers starting at $(x, y) = (500, 500)$, each taking a maximum of 150000 steps, the distribution shown in Figure 2 was generated.
Over these 40 walks, only 10 walkers 'escaped' the bounds of the 1001×1001 grid, and the average speed at escape (across the 10 walkers) was $\overline{v} \approx 7.749 \times 10^{-3}$ units per step. In comparison with the average speed value from the first simulation, this correlates with a 2 order of magnitude reduction in average speed for a two order of magnitude increase in grid length. More tests could support whether this inverse trend holds.

Given that an interaction with a barrier didn't necessarily *halt* the walker, the velocity of the walkers at escape shouldn't have been affected too much. This is because, as was said before, the direction of walker movement was redrawn *in the same step* until it was permitted to move again.

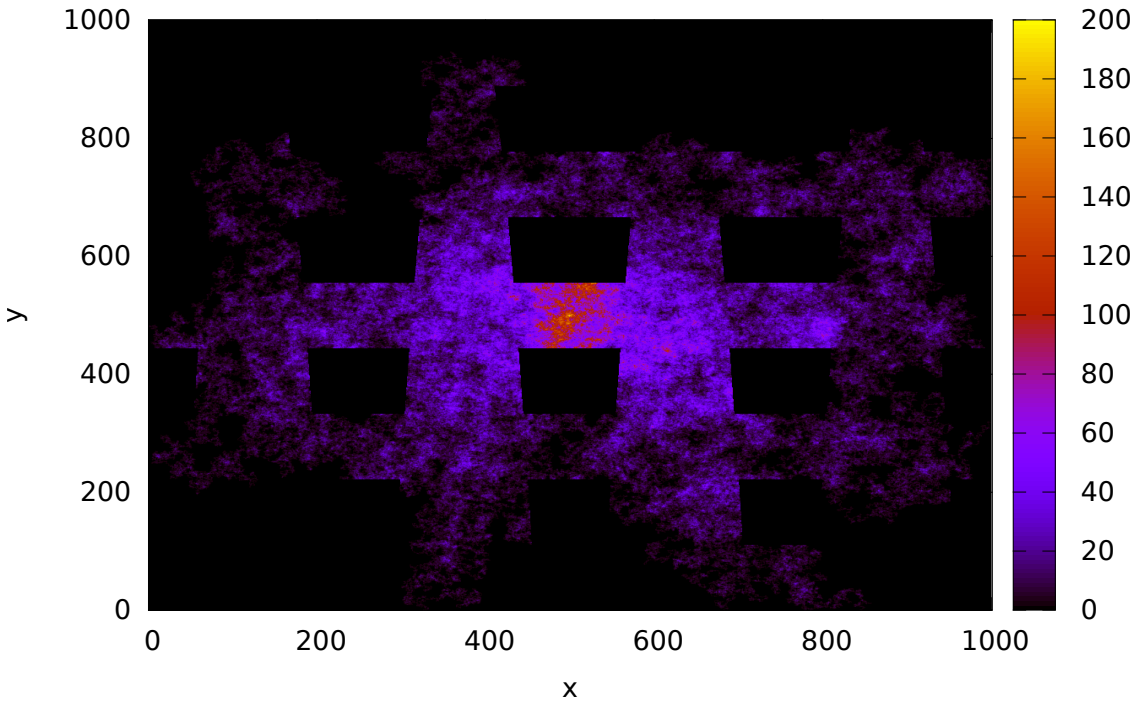

Figure 2: Distribution of Random Walkers with Barriers

Bar the clear evidence of voids from the barriers in the above heatmap, the distribution of walkers once again resembles that of a two-dimensional Gaussian as in the first simulation. This further supports the claimed random nature of the walkers.