Solving Quantum Systems Numerically

Ewan Saw
CID: 01521087

***Abstract*—In this investigation, we explore different numerical integration methods that can be applied to solve for the probability of a particle existing in a given volume within a quantum-mechanical system. The methods used were The Extended Trapezoidal Rule, Extended Simpson's Rule and Monte Carlo integration methods. Such methods included using uniform and linear importance sampling. Using these, we solve for the probability of 3 quantum states: a 1D ground state wavefunction, a 3D ground state wavefunction, and a state with one unit of orbital angular momentum. The final probabilities were found to be 0.497661132, 0.123254040 and 0.118135520 respectively. Their relative errors were 3.3306691 x 10^{-16}, 4.1256560 x10^{-9} and 4.12565626 x 10^{-9}.**

I. Introduction

Within a quantum-mechanical system, it is often helpful (and sometimes required) for physicists to be able to determine the positions of particles present at a particular time. Such an exercise allows us to gain further insight into the workings of the system as well as the particle's wavefunction.

Due to the uncertainty principle and the nature of quantum mechanics, the exact position of a particle within such a system cannot be exactly identified. Instead, we are able to evaluate the probability of the particle existing within a specific volume of space. This is achieved by integrating the absolute value of the particle's wave function squared over the desired volume.

In practice, analytical solutions to such integrands are either difficult to determine or simply do not exist. As such, it is useful to equip a numerical approach in solving for these probabilities. This allows us to obtain solutions independent of the forms of the particles' wavefunctions.

In this investigation, we will explore a variety of numerical methods that can be applied in solving these systems: first in the 1D case, before generalising our methods to higher dimensions. We will also consider the efficiencies of such methods and make comparisons between the relative accuracies of our results.

II. Theory

To begin, we consider the wavefunction of a particle in a 1D quantum-mechanical system $\psi(x,t)$. Assuming it is time-independent, the ground-state wavefunction ψ_0 of this particle in a simple harmonic oscillator potential can be written

$$\psi_0(x) = \left(\frac{m\omega}{\hbar\pi}\right)^{\frac{1}{4}} e^{-m\omega x^2/2\hbar}, \tag{1}$$

where ω is the natural angular frequency of the harmonic oscillator.

The probability of finding the particle within the range $x_1 = 0$ to $x_2 = 2\sqrt{\hbar/m\omega}$ is then given by

$$P = \int_0^{2\sqrt{\hbar/m\omega}} \mid \psi_0(x) \mid^2 dx. \tag{2}$$

We can then simplify equation 2 significantly using the substitution $u = x\sqrt{m\omega/\hbar}$. Hence

$$P = \int_0^2 \sqrt{\frac{1}{\pi}} e^{-u^2} du, \tag{3}$$

showing that evaluating the probability of such a particle within these limits is independent of ω and its mass m.

The following sub-sections will briefly describe the numerical methods used to solve equation 3, 14 and 16. Details of their respective implementations in the code are then outlined in section III.

A. Newton-Cotes Methods

Newton-Cotes integration methods involve dividing an integral into a number of equally spaced sub-regions, with a constant step-size h. These regions are then independently integrated over to approximate the result[2].

1) Extended Trapezoidal Rule: The simplest approximation of an integral involves linearly interpolating between two sample points x_i and x_{i+1} using the Trapezoidal rule. Instances of this approximation can then be stitched together to approximate an entire integral, i.e. the *Extended Trapezoidal Rule* [2],

$$\begin{aligned}\int_{x_0}^{x_{n-1}} f(x)dx = \quad & h[\tfrac{1}{2}f(x_0) + f(x_1) + ... \\ & +f(x_{n-2}) + \tfrac{1}{2}f(x_{n-1})] \\ & +\mathcal{O}\left(\tfrac{(x_{n-1}-x_0)^3}{N^2}\tfrac{d^2f}{dx^2}\right).\end{aligned} \tag{4}$$

Here, the step-size $h = (x_{n-1} - x_0)/N$, where N is the total number of steps. Note that the last term in equation 4 approximates the error of this method to $\sim \mathcal{O}\left(h^2 \frac{d^2f}{dx^2}\right)$.

2) Extended Simpson's rule: Building on the Extended Trapezoidal, the *Extended Simpson's Rule* allows for quadratic interpolation between samples to approximate the integral. This is given by[2]

$$\begin{aligned}\int_{x_0}^{x_{n-1}} f(x)dx = \quad & h[\tfrac{1}{3}f(x_0) + \tfrac{4}{3}f(x_1) + \tfrac{2}{3}f(x_2) \\ & +\tfrac{4}{3}f(x_3) + ... + \tfrac{2}{3}f(x_{n-3}) \\ & +\tfrac{4}{3}f(x_{n-2}) + \tfrac{1}{3}f(x_{n-1})] \\ & +\mathcal{O}\left(\tfrac{(x_{n-1}-x_0)^5}{N^4}\tfrac{d^4f}{dx^4}\right).\end{aligned} \tag{5}$$

Again, the last term in equation 5 represents the error in this method and is $\sim \mathcal{O}\left(h^4 \frac{d^4f}{dx^4}\right)$. This shows that this method is $\mathcal{O}(h^2)$ more accurate than the Extended Trapezoidal Rule.

B. Monte Carlo Integration

Monte Carlo methods employ the generation of random numbers in order to statistically solve problems. In the case of integration, it is commonly used in performing integrals over many variables and dimensions[2].

For a d-dimensional function $f(\vec{x})$, its integral over a d-dimensional volume V can be approximated by randomly taking N samples of the function within the volume of integration. By making use of the relation between the mean value of a function and its integral over a specific volume, we can estimate the integral $\hat{I}$.

$$\int_V f(\vec{x})d\vec{x} \approx \hat{I} = \frac{V}{N}\sum_{i=1}^{N} f(\vec{x}_i) \pm \frac{V}{\sqrt{N}}\sigma_{f_i}, \tag{6}$$

where σ_{f_i} is an estimate of the standard deviation of the samples

$$\sigma_{f_i}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(f(\vec{x}_i) - \langle f\rangle)^2. \tag{7}$$

Note that the error in Monte Carlo integration scales as $\mathcal{O}(h^{1/2})$. i.e. the error scales as the square root of the number of samples[1].

The case where we pick our samples from a uniform PDF is known as Uniform Sampling. In addition to this, we also consider a more efficient sampling method.

1) Importance Sampling: In a number of situations, it is usually more efficient to evaluate an integral with Monte Carlo by biasing the sampling of the integrand to regions where it has more 'weight'[2]. To do this, we can write the integral as

$$I = \int_V \frac{f(\vec{x})}{P(\vec{x})}P(\vec{x})d\vec{x} = \int_V Q(\vec{x})P(\vec{x})d\vec{x} = \langle Q\rangle, \tag{8}$$

where

$$\langle Q\rangle = \frac{1}{N}\sum_{i=1}^{N} Q(\vec{x}_i) \pm \frac{1}{\sqrt{N}}\sigma_{Q_i}. \tag{9}$$

Here, $Q = f(\vec{x})/P(\vec{x})$ and $P(\vec{x})$ is the PDF of the sampling rate we desire. Note that this also means that $P(\vec{x})$ must be normalised. The variance of Q_i is then given by

$$\sigma_{Q_i}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Q(\vec{x}_i) - \langle Q\rangle)^2. \tag{10}$$

Using this, we can estimate the standard deviation of our results from importance sampling.

2) Transformation Method: In order to generate random deviates of y which are distributed according to a sampling PDF $P(\vec{x})$, we can apply the Transformation method to random deviates of x uniformly generated between 0 to 1.

The method entails finding the cumulative distribution function $F(y)$ of $P(y)$ and inverting it to find a relationship between the deviates[2], i.e.

$$F(y) = \int_{-\infty}^{y} P(\tilde{y})d\tilde{y}. \tag{11}$$

Inverting this,

$$y = F^{-1}(x), \tag{12}$$

such that a uniform deviate x plugged into equation 12 produces deviates y generated according to $P(y)$.

C. Higher Numbers of Dimensions

Generalising our quantum-mechanical system into a 3D space, the wavefunction for a given particle then becomes $\Psi(x, y, z)$. In this investigation, we consider the probability integrals of a particle in two different 3D states.

1) Ground State: The overall ground state of a particle in a 3D simple harmonic oscillator potential is given by the product of its 1D ground states

$$\Psi(x, y, z) = \psi_0(x)\psi_0(y)\psi_0(z). \tag{13}$$

We wish to find the resulting probability of finding this particle within a cube spanning from 0 to $2\sqrt{\hbar/m\omega}$ in each dimension. By applying the same substitution to the integral of equation 13 as we did for equation 2, we obtain the integral

$$P = \iiint_0^2 \left(\frac{1}{\pi}\right)^{3/2} e^{-x^2-y^2-z^2}\, dx\, dy\, dz. \tag{14}$$

2) One Unit of Orbital Angular Momentum: In 3D, particles are able to possess an orbital angular momentum. A state with one unit of orbital angular momentum can be written as

$$\Psi_1 = \frac{1}{\sqrt{2}}[\psi_1(x)\psi_0(y) + i\psi_0(x)\psi_1(y)]\psi_0(z). \tag{15}$$

Following the same process as before, the resulting probability integral over the same cube volume for this state is then given by

$$P = \iiint_0^2 \left(\frac{1}{\pi}\right)^{3/2} (x^2+y^2)e^{-x^2-y^2-z^2}\, dx\, dy\, dz. \tag{16}$$

Now, we are able to apply our methods to numerically integrate equations 3, 14 and 16.

III. Implementation of Methods

This section outlines the steps and key points involved in implementing the methods described in section II. For every function defined in the code, an additional user-specified argument for a desired relative accuracy ϵ was added. Each routine then refines the relative accuracy of the result to fit this value.

Vectorisation of Code

When attempting to achieve results with high orders of accuracy, the methods we use below are often liable to the efficiencies of our routines as large numbers of function evaluations are often required. As opposed to using loops to perform a large number of operations in series, *numpy* allows us to perform operations on entire arrays of values in parallel, known as *vectorisation*. This has been shown to greatly reduce time execution of routines in Python[3]. As such, vectorisation in our implementation was used where appropriate.

A. Newton-Cotes

1) Extended Trapezoidal ('trap_int' Function): Constructing an algorithm which refines the relative accuracy of results produced by this method simply involves increasing the number of samples until a desired accuracy is reached. This can be achieved by a systematic reduction of the step size h after each iteration.

The algorithm begins with a step size of $h = \frac{(x_{n-1}-x_0)}{N}$, where $N = 1$. This evaluates the values of the integrand at the endpoints, estimating the integral using equation 4. The following iterations then calculate the values of the function at the midpoints between every previously evaluated point. Essentially, this allows us to continuously iterate without having to evaluate the integrand at the same point twice.

Further inspection of equation 4 allows us to find an explicit recurrence relation of the process described above. Between the j and $j+1$ terms[2],

$$T_{j+1} = \frac{1}{2}T_j + \frac{1}{2}h_j \sum_{i=1}^{2^{j-1}} f\left(x_0 + \frac{2i-1}{2}h_j\right), \tag{17}$$

where the step size is halved after each iteration,

$$h_{j+1} = \frac{h_j}{2}. \tag{18}$$

Here, T_1 is the first iteration where only 2 samples are evaluated, i.e. the start and end points.

The routine applies equation 12 and 13 recursively within a *while* loop to iterate through results of increasing accuracy. After each iteration, it checks for the convergence of the results[2],

$$\left|\frac{T_{j+1}-T_j}{T_j}\right| < \epsilon. \tag{19}$$

If equation 19 is satisfied, T_j is returned.

2) Extended Simpson's ('simp' Function): This algorithm applies the same principles used for the implementation of the Extended Trapezoidal rule. Simpson's Rule is constructed such that the order h^3 and h^4 errors obtained from the Trapezoidal Rule cancel. This allows us to obtain the j^{th} iteration of the Simpson's Rule by re-weighting and combining two successive iterations of the Trapezoidal Rule[2],

$$S_j = \frac{4}{3}T_{j+1} - \frac{1}{3}T_j. \tag{20}$$

Considering the convergence criterion for this method, an inequality with a similar form to equation 19 can be found. Applying equation 20, we find

$$\left|\frac{S_{j+1}-S_j}{S_j}\right| = \left|\frac{4T_{j+2}-T_{j+1}}{4T_{j+1}-T_j} - 1\right| < \epsilon. \tag{21}$$

S_j is then returned if equation 21 is satisfied.

Newton-Cotes in Higher Dimensions

3) 'trap_3d' & 'simp_3d' Functions: In order to integrate equations 14 and 16, we need to generalise our methods to 3D. Consider a wavefunction $\Psi(x,y,z)$ to be integrated over a 3D volume V. We can write

$$I = \iiint_V \Psi(x,y,z)\,dx\,dy\,dz = \int g(x)dx, \tag{22}$$

where

$$g(x) = \int h(x,y)dy, \tag{23}$$

and

$$h(x,y) = \int \Psi(x,y,z)dz. \tag{24}$$

This can be interpreted as applying our 1D integrator once in each dimension. As a result, by nesting our 1D Newton-Cotes integrators within each other, we are able to construct a 3D integrator. This was done for both Trapezoidal and Simpson's rules.

B. Monte Carlo

1) Uniform Sampling: Following the routine described in section II-B and utilising equation 6, we wrap this method within a *while* loop that continuously draws random samples of $\vec{x}$ from a uniform PDF. $\hat{I}$ is then calculated using equation 6.

2) Importance Sampling: As described in section II-B1, the routine follows the same structure as the Uniform Sampling case. This uses equations 9 and 10 to calculate the integral and its uncertainty respectively.

In this investigation, the choice of sampling PDF $P(\vec{x})$ was required to be of the form

$$P(x) = Ax + B, \tag{25}$$

such that the choice of constants gave a linear equation that dropped from a maximum x_1 to a non-zero minimum x_2. In order to generate samples according to equation 25, the *Transformation Method* was used.

Applying equation 12 to equation 25, we find

$$y = -\sqrt{\frac{2}{A}\left(x + \frac{B^2}{2A}\right)} - \frac{B}{A}. \tag{26}$$

This transforms random numbers generated by the *'numpy.random.random'* function into deviates distributed by $P(\vec{x})$. Given this, we are able to calculate the integral with the same process as before using equation 9.

Determining Convergence: Expanding equation 10 for the variance of our samples, we can show that

$$\sigma^2_{Q_i} = \frac{\sum Q^2(\vec{x}_i) - 2\sum Q(\vec{x}_i)\langle Q\rangle + N\langle Q\rangle^2}{N-1}. \tag{27}$$

Equation 27 then allows us to evaluate the variance and hence the error by storing *only* the sums and squared sums of the evaluated samples, as opposed to storing the entire set of values. This in turn saves a significant amount of memory.

Note that equation 27 applies to both types of sampling, but it is worth pointing out the extra factor of V present in the error for equation 6 as opposed to equation 9.

Now, we are able to determine the convergence criterion for Monte Carlo integration. As we require the result to satisfy a relative accuracy ϵ, convergence occurs when

$$\left|\frac{error}{\hat{I}}\right| < \epsilon. \tag{28}$$

When this is satisfied, the routine breaks out of the *while* loop and returns the most recent value of $\hat{I}$ and its error.

Monte Carlo in Higher Dimensions

Due to the nature of Monte Carlo methods, generalising them to higher dimensions requires less work. It can be seen that equations 6 and 9 apply to vectors, hence we are able to integrate up to d-dimensions using Monte Carlo without modifying the original methods.

In the importance sampling case, we also recognise that $P(\vec{x})$ can be separated into the product of the PDF in each dimension. For 3D,

$$P(\vec{x}) = P(x)P(y)P(z). \tag{29}$$

IV. Results and Analysis

A. Newton-Cotes Integration in 1D

Equation 3 was solved using our Newton-Cotes methods described in previous sections, giving Table I.

TABLE I
Comparison of Newton-Cotes method results with a desired relative accuracy $\epsilon = 10^{-6}$ for 1D ground state.

	Trapezoidal Rule	Simpson's Rule
Integration Estimate	0.497661	0.497661
Relative Error	3.1683×10^{-7}	6.6019×10^{-8}
Number of Samples	256	32
Execution Time (s)	0.002058	0.000568

It can be seen that for a desired relative accuracy of $\epsilon = 10^{-6}$, both methods gave results with an agreement of up to 6 significant figures. Their relative errors also show that they scale well beyond the desired accuracy. It is also apparent that the Simpson's Rule outperforms the Trapezoidal rule in every aspect as the relative error, number of samples required, and execution time are all an order of magnitude lower than the Trapezoidal Rule.

This is in agreement with the errors outlined in equations 4 and 5. Comparing the two, the errors for the Extended Simpson's rule scale with an extra factor of $1/N^2$, which shows why the result for the Simpson's Rule converges at a much faster rate.

B. Monte Carlo Integration in 1D

Applying our Monte Carlo methods to the same integral, we obtain Table II. The routine used was able to compute the integrals up to a relative accuracy of 1×10^{-5} for both types of sampling within a reasonable time. However, we were unable to reach accuracies of 10^{-6}.

TABLE II
Comparison of Monte Carlo method results with number of samples for 1D ground state.

Sampling	Relative Accuracy	No. of Samples	Result
Uniform	1×10^{-4}	1.6110×10^{8}	0.49765
Linear	1×10^{-4}	1.1600×10^{7}	0.49762
Uniform	1×10^{-5}	1.6107×10^{10}	0.49766
Linear	1×10^{-5}	1.1586×10^{9}	0.49766

From equation 6 and 9, we know the the error scales as $N^{-1/2}$ with an additional proportionality constant. By fitting a function of this form to the relative accuracies with respect to the number of samples evaluated, we can estimate the total number of evaluations required to reach a specific relative accuracy. The fit was found to give the relation

$$\epsilon = 1.2692N^{-\frac{1}{2}}, \tag{30}$$

for the uniform sampling case. Using this relation, we estimate the number of evaluations required to reach $\epsilon = 1 \times 10^{-6}$ to be $N \sim 1.611 \times 10^{12}$. The same was done for the linear sampling case and $N \sim 1.158605 \times 10^{11}$.

The results summarised in Table II validate our intuition wherein using importance sampling would improve the efficiencies of our routine and allow us to converge to the true result at a faster rate. Fig. 1 gives a good representation of how both sampling methods compare in terms of reaching convergence.

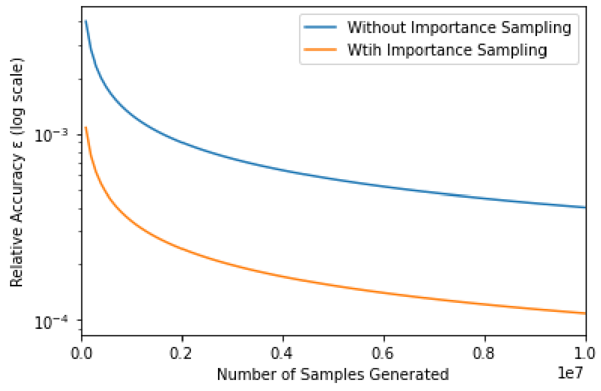

Fig. 1. Plots of how relative accuracy of Monte Carlo methods vary with the number of samples generated using different sampling methods, with the y-axis on a log scale. It can be seen that samples generated using importance sampling (orange) converge to a lower error much quicker than the uniform sampling method (blue).

Error Scaling of Methods

Further inspection of the errors in equations 4, 5 and 9 allow us to compare how the error scales with the number of samples for each method. As previously mentioned, we know that the trapezium rule has a truncation error that goes as $\sim \mathcal{O}\left(h^2\right)$ and $\sim \mathcal{O}\left(h^4\right)$ for Simpson's rule. It can also be shown that

$$h \sim N^{-1/d}, \tag{31}$$

where d is the number of dimensions. As such, we can predict how the error scales for each method in any number of dimensions[2], as shown in Table III. Note that due to how the Monte Carlo methods are constructed, its error scales at the same rate regardless of the number of dimensions it is being applied to.

C. 3D Ground State Integration

Equation 13 was integrated using 3D generalisations of all methods. A relative accuracy of $\epsilon = 10^{-4}$ was chosen as it

TABLE III
COMPARISON OF ERROR SCALING OF METHODS IN INCREASING DIMENSIONS.

d	**Trapezoidal**	**Simpson's**	**MC**
1	N^{-2}	N^{-4}	$N^{-1/2}$
2	N^{-1}	N^{-2}	$N^{-1/2}$
3	$N^{-2/3}$	$N^{-4/3}$	$N^{-1/2}$
4	$N^{-1/2}$	N^{-1}	$N^{-1/2}$

was the minimum accuracy that all methods could achieve relatively quickly. We are able to make sense of the relative execution times of the results summarised in Table IV by referring to Table III. Table III shows that in 3D, the rate of convergence for a given number of samples scales quickest for Simpson's rule, followed by Trapezoidal, and finally Monte Carlo. We have also concluded that Linear Sampling converges to a given solution faster the uniform sampling. All of this agrees with the results we have obtained in this section. In some ways, it validates our understanding of the different methods and instills confidence in our implementation of the code.

TABLE IV
RESULTS OF 3D GROUND STATE INTEGRAL AT A DESIRED RELATIVE ACCURACY OF $\epsilon = 10^{-4}$ FOR ALL METHODS.

Method	Relative Accuracy	Execution Time (s)	Result
Trapezoidal	8.0854476×10^{-5}	0.7427649	0.1232142
Simpson's	1.6876497×10^{-5}	0.1130981	0.1232474
MC Uniform	9.9999787×10^{-5}	328.2860391	0.1232546
MC Linear	9.9926124×10^{-5}	2.0438957	0.1232469

D. 3D Orbital Angular Momentum

Having compared the relative execution times and efficiencies of our different methods, we conclude that the Extended Simpson's Rule outperforms the other methods for evaluating 3D integrals. As such, this was used to evaluate equation 16. This gave

$$P = 0.11813552, \epsilon = 4.12565626 \times 10^{-9}. \quad (32)$$

The same was done for all previous integrals to obtain results with the highest degrees of accuracy using Simpson's. The result for the 1D ground state intuitively makes sense as the wavefunction is mainly localised at the origin. Hence the probability of finding it from 0 to 2 is close to a half. The same reasoning applied for the 3D ground state. As it is in 3D, the probability would be $\sim 0.5^3 = 0.125$. Whilst a particle with orbital angular momentum is much more likely to be delocalised from the origin due to it possessing a higher energy.

TABLE V
SUMMARY OF FINAL RESULTS USING SIMPSON'S RULE.

Integral	Relative Accuracy	Result
1D Ground State	$3.3306691 \times 10^{-16}$	0.497661132
3D Ground State	4.1256560×10^{-9}	0.123254040
1 Unit Orbital Ang. Mom.	$4.12565626 \times 10^{-9}$	0.118135520

Final Validations

1) Importance Sampling: As a final check to validate our methods, we consider the generation of our sample points for our importance sampling case in MC integration. We wish to draw samples from a sampling PDF that follows the general shape of the integrand. As such, Fig. 2 shows that we manage to achieve this using the transformation method as the distribution of samples lines up perfectly with our sampling PDF. Intuitively, this shows that we are sampling the function more at points where the integrand has more 'weight'.

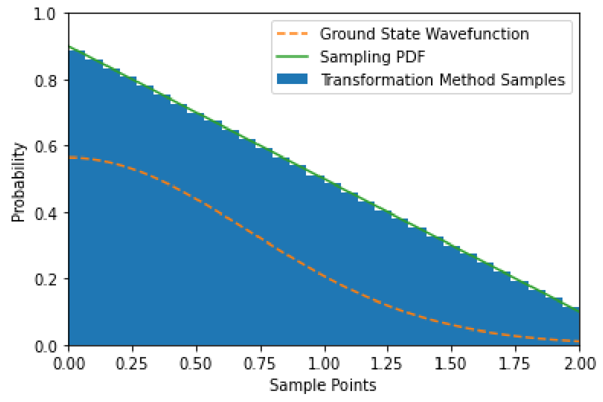

Fig. 2. Distribution of sample points generated using the Transformation method (blue), with the ground state wavefunction (orange) plotted over it. The histogram shows that the samples generated by this method line up with the desired sampling PDF (green), and that the sampling rate decreases with increasing x-values.

2) Distribution of Monte Carlo Results: We can examine the spread of the results obtained from our Monte Carlo methods by taking multiple cycles of our method and plotting the distribution on a histogram, as shown in Fig. 3.

The peak and mean of our Gaussian seems to be centred along the true value of the integral we are estimating. This in turn validates our method. This is because when we have evaluated a sufficiently large number of cycles, we are be able to converge on the true result with Monte Carlo integration.

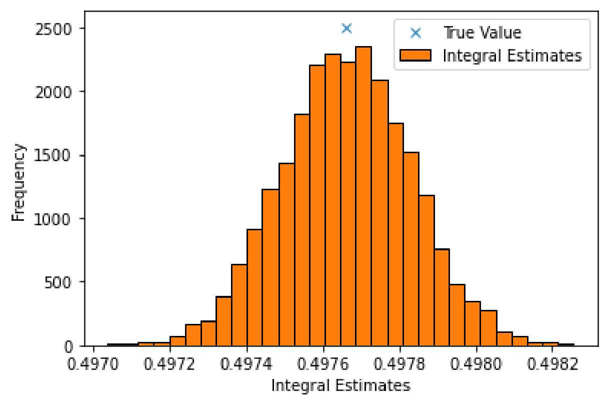

Fig. 3. Distribution of integral estimates generated by our MC method with linear sampling. The method was used to integrate over the 1D ground state wavefunction over 10000 cycles. The histogram resembles a Gaussian with its peak centred close to the true value of the integral. This true value was obtained from Wolfram Alpha.

V. Conclusion

Over the course of this investigation, we explored a variety of numerical methods that could be applied in solving for the probabilities of quantum particles existing in particular regions of space, namely Newton-Cotes and Monte Carlo methods. By closely inspecting the theory of these methods, we were able to make valuable comparisons with regards to their efficiencies as well as their pitfalls.

By considering how these functions operate, we were able to identify key choices that allowed us to implement our code efficiently, as well as ensuring they were optimised. This was paramount, as attempting to evaluate the integrals to high degrees of accuracy involved function evaluations up to the order of tens of billions. This optimisation is believed to mainly stem from the vectorisation of our code.

Having evaluated these integrals according to specific relative accuracies, we were able to analyse the difference in execution time and number of samples required by each method to converge. After thorough analysis of the theory and error truncation of each method, we were able to conclude the the Extended Simpson's Rule was the most efficient method for calculating integrals from 1D to 3D. However, when extending up to higher dimensions, Monte Carlo methods are much more viable. We were also able to validate our code by checking the histograms of distributed points and results.

The results for each integral were then reproduced to the highest achievable orders of accuracy using Simpson's and are summarised in Table 5.

References

[1] Patrick Hanbury. *Monte Carlo Simulations with Python (Part 1).* https://towardsdatascience.com/monte-carlo-simulations-with-python-part-1-f5627b7d60b0. 2019.

[2] Paul Dauncey Mark Scott. *Computational Physics Course Notes.* Imperial College London. 2020.

[3] Rakesh Patidar. *Vectorization in Python.* https://www.geeksforgeeks.org/vectorization-in-python/. 2019.