Solving 1D Quantum Systems with Numerical Methods

Yaojia Huang & 01213992

December 17, 2018

Abstract

This report shows the implementations of Newton-Coates integration, Monte Carlo integration and VEGAS algorithm, which can be applied to calculate probabilities in 1D quantum systems. The result shows that VEGAS algorithm and methods for the trapezoidal and Simoson's rules achieved exponential convergence, but the convergence rate of Monte Carlo methods is proportional to the square root of the sample size.

1 Introduction

When dealing with quantum systems, it could be extremely difficult, or even impossible, to analytically integrate the probability density function (PDF) of the particles. Fortunately, many numerical methods have been developed to calculate integrals with high accuracy. For example, Newton-Coates integration, named after Sir Isaac Newton and Roger Cotes, estimates the integral by summing the areas between closely spaced intervals. Moreover, Monte Carlo (MC) methods can achieve the same goal by taking random samples, but they usually spend much longer time than Newton-Coates integration in order to reach a desired accuracy. In this project, two kinds of Newton-Coates integration, the trapezoidal rule and Simpson's rule, were applied to compute the integral; then different MC approaches (naïve Monte Carlo (NMC) integration and MC integration with importance sampling (IS)) were adopted and optimised. Finally, the VEGAS algorithm, a method for adaptive stratified sampling, was also implemented. The methods of implementing these algorithms are demonstrated and the accuracies of the results are compared in this report.

2 Theory

2.1 Trapezoidal rule and Simpson's rule

The trapezoidal rule is a technique that evaluates the integral of a given function $f(x)$ by sampling $n+1$ evenly spaced points $x_0, x_1, x_2, ..., x_n$ and summing every trapezoidal area between adjacent partitions shown in Figure 1a. Simpson's rule, on the other hand, requires three sample points located at x_i, x_{i+1}, x_{i+2} to calculate each sub-area because the upper boundary of the sub-region is approximated by a quadratic function (Figure 1b). Simpson's rule can produce more accurate result than the trapezoidal rule because it generates a better fit for the original function.

The equations for the trapezoidal rule and Simpson's rule for a single sub-region are, respectively,

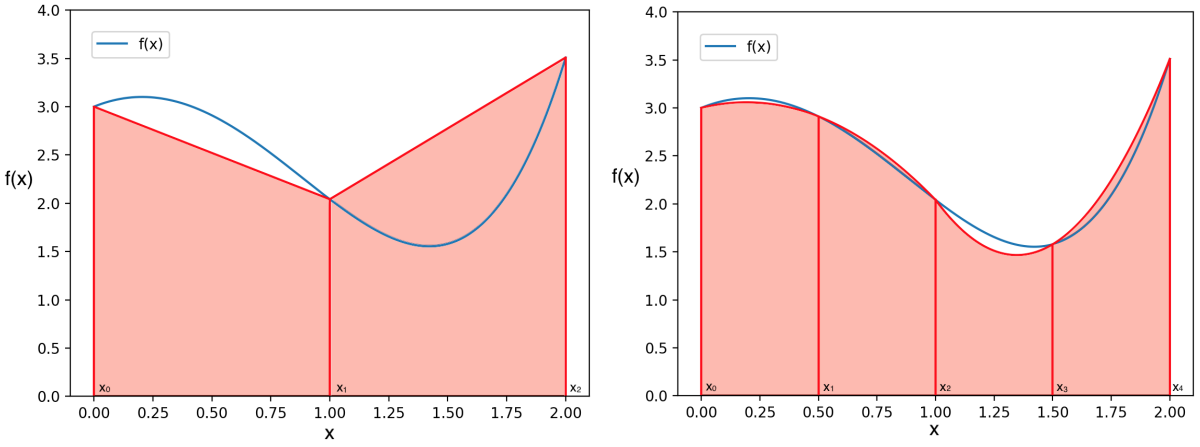
$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}\Big[f(x_i) + f(x_{i+1})\Big] + O\Big(h^3\frac{d^2f}{dx^2}\Big), \tag{1}$$

$$\int_{x_i}^{x_{i+2}} f(x)dx = \frac{h}{3}\Big[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\Big] + O\Big(h^5\frac{d^4f}{dx^4}\Big), \tag{2}$$

where h is the horizontal interval between adjacent points. The extended trapezoidal and Simpson's rules can be obtained by summing up all sub-regions:

$$\int_{x_0}^{x_n} f(x)dx \approx \frac{h}{2}\Big[f(x_0) + 2f(x_1) + 2f(x_2) + ... + 2f(x_{n-1}) + f(x_n)\Big], \tag{3}$$

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx \approx \frac{h}{3}[&f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + ... \\ &+ 4f(x_{n-3}) + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)].\end{aligned} \tag{4}$$



(a) Trapezoidal Rule **(b)** Simpson's Rule

Figure 1: Visualisation of Trapezoidal and Simpson's Rules. The trapezoidal rule uses straight lines to approximate the shape of the original function, while Simpson's rule uses quadratic functions to do so. The red shaded region represents the estimated area.

2.2 Naïve Monte Carlo Integration

The NMC integration is done by randomly selecting points whose x-coordinates are bounded by $[x_0, x_N]$ and y-coordinates bounded by $[0, max(f(x))]$, and accepting only the points that fall below $f(x)$ (Figure 2). Denote the number of accepted points as N_{acc} and the total number of points as N, the integral can be approximately evaluated by taking the product of the area of the bounded region and N_{acc}/N. According to the law of large numbers, the result approaches the actual value as N grows larger.

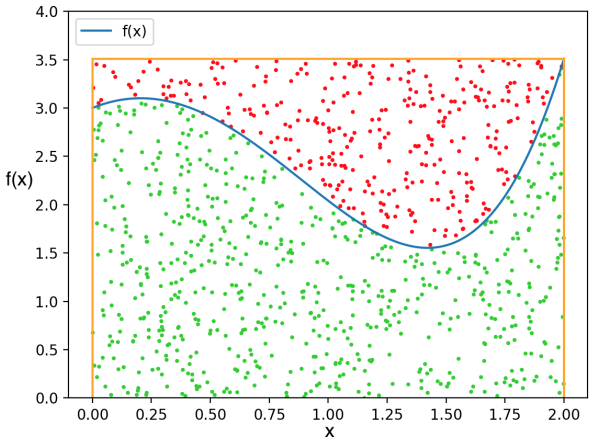


Figure 2: Demonstration of the NMC Method. The randomly chosen points are accepted(marked green) if they fall below $f(x)$; otherwise rejected(marked red).

2.3 Monte Carlo Integration with Importance Sampling

In most cases, uniform sampling is not a good practice; the result could converge much faster if the samples were to be picked from a more appropriate distribution, $p(x)$, so that more samples are drawn from the region where $f(x)$ makes larger contribution to the integral. Such method is called importance sampling.

The integral of $f(x)$ can be alternatively expressed as

$$\int_{x_0}^{x_t} f(x)dx = \int_{x_0}^{x_t} \frac{f(x)}{p(x)} p(x)dx \tag{5}$$

$$= \left\langle \frac{f(x)}{p(x)} \right\rangle. \tag{6}$$

The integral can be computed by sampling $x_i \sim p(x)$ and taking the average of $f(x_i)/p(x_i)$.

The best PDF is directly proportional to $f(x)$. Denote the exact integral as I, the optimal PDF is $F(x) = f(x)/I$; therefore, $f(x_i)/F(x_i)$ produces I for any x_i and the variance is always 0.

2.4 Monte Carlo Integration with Stratified Sampling

Stratified sampling is a common way to reduce the standard error of the result in an MC simulation [1]. It is a process that divides the integration region into strata whose total area is 1, and takes samples from each stratum. Denote the area of stratum j as A_j, then the number of random samples drawn in stratum j is $N_j = NA_j$, where N is the total number of samples.

3 Methods

3.1 Trapezoidal Rule

To estimate the integral of $f(x)$ using the trapezoidal rule, the adopted procedure is to divide the integration region further and further until the percentage change of the area between two successive divisions is less than the user-specified relative accuracy.

Suppose at the k-th iteration, the estimated area is T_k and the locations where the integration region is divided are $x_0, x_2, x_4, ..., x_{2^k}$. If the relative accuracy, $|\frac{T_k - T_{k-1}}{T_{k-1}}|$, is larger than the one defined by the user, the region is further divided at the middle point between every pair of adjacent partitions. By doing so, the partition separation, h, is halved. Then, according to Equation 3, T_{k+1} can be calculated by accounting for the reduced separation and contributions from $f(x_1), f(x_3), ..., f(x_{2^k-1})$.

The advantage of this method is that the information of the result obtained in one iteration can be used in the next iteration so that the program does not need to re-evaluate $f(x)$ at every x_i. In addition, the result converges to the exact value exponentially because the number of divisions is doubled in every iteration.

3.2 Simpson's Rule

The method for the trapezoidal rule can be rehashed for the implementation of Simpson's rule. According to Equation 3 and 4, the area computed in the k-th iteration using Simpson's rule, S_k, can be solved with the results from the trapezoidal rule:

$$S_k = \frac{4}{3} T_{k+1} - \frac{1}{3} T_k. \tag{7}$$

Using this equation, the method can be implemented and it does only one more calculation in each loop than the trapezoidal method. The error terms of order h^3 and h^4 are cancelled, which gives a more accurate result.

3.3 Naïve Monte Carlo Integration

The NMC integration is implemented as described in Section(2.2.1). The sampling process is constructed in a loop, which terminates when the relative standard error, ϵ, of the result is less than the user input. Since the method is a binomial process, ϵ is calculated by $\sqrt{\frac{p(1-p)}{N}}/\bar{I}$, where p is approximated by N_{acc}/N and $\bar{I}$

is the estimated integral.

In the program, a function that sets the minimum number of samples to be drawn is implemented. This function reduces the risk that the program terminates too early because ϵ is not monotonically decreasing (e.g. ϵ is 0 when there is only one sample). Moreover, the condition for termination is that ϵ must be less than the targeted error for a certain number of times consecutively so that the program does not terminate when ϵ fluctuates occasionally below the targeted error. These implementations were also adopted in other MC methods.

3.4 Monte Carlo with Importance Sampling

To implement IS, the program requires the user to input a sampling PDF, $p(x)$. The cumulative probability density function (CDF) of $p(x)$ could be either invertible or not. If the CDF is invertible, the user could provide the inverted function to generate sample points, which is more efficient. When the inverted CDF is not given, the program would select samples using rejection method with a constant comparison function that lies above the maximum value of $p(x)$ within the integration region. The maximum is found using simulated annealing, in which new value of $p(x)$ is accepted if it is larger than the previous one, or rejected with probability according to Boltzmann probability distribution when it becomes smaller. Normally the maximum is obtained when the temperature reaches zero. The searching process is terminated if no new value is accepted after a certain number of consecutive iterations to avoid dead loops. This algorithm is adopted because it could escape from local maxima.

For every sample x_i, the values of $\frac{f(x_i)}{p(x_i)}$ and $\left(\frac{f(x_i)}{p(x_i)}\right)^2$, denoted as I_i and I_i^2, are calculated. The average of the former, $\bar{I}$, gives estimation of the integral (Equation 6) and the latter is used to compute ϵ of the result. The equation for ϵ is

$$\epsilon = \sqrt{\frac{\langle I_i^2 \rangle - \bar{I}^2}{N}} \Big/ \bar{I}. \tag{8}$$

In every loop, ϵ is calculated and compared with the targeted relative error. The condition for termination is the same as that described in Section(3.3).

3.5 Optimising Probability Density Function for Importance Sampling

The best PDF for IS is, as mentioned, $F(x) = f(x)/I$, but it is impossible to get this PDF beforehand because it requires the knowledge of I. However, $F(x)$ can be approximated using kernel density estimation(KDE).

The first step of KDE is randomly selecting N samples on the x-axis according to $f(x)$ using the rejection method. Let each sample represent a kernel function centred at x_i, the PDF of $f(x)$ can be obtained by normalising the summation of all kernel functions. The mathematical expression for the PDF is

$$p(x) = \frac{1}{Nb} \sum_{i=1}^{N} K\left(\frac{x - x_i}{b}\right), \tag{9}$$

where K is the kernel function and b is the width of K. In this project, b is determined by the rule of thumb [2] and the kernel function is the standard normal distribution:

$$K(x') = \frac{1}{\sqrt{2\pi}} e^{\frac{-x'^2}{2}}. \tag{10}$$

KDE does not work properly when dealing with a function in a bounded region because the estimated PDF would extend beyond the boundaries, which causes probability overflow (Figure 3). In this situation, the estimation is no longer accurate.

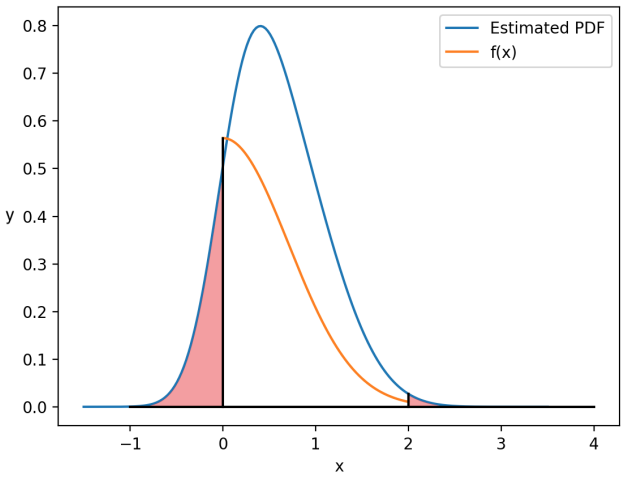


Figure 3: Probability Overflow Produced by KDE. KDE is used to estimate the PDF of $f(x)$ within $[0, 2]$, whereas the resultant PDF is not restricted in the bounded region. The red shaded region represents the overflowed probability

To resolve this problem, the finite region needs to be mapped to $(-\infty, \infty)$. The new coordinate system is chosen to be

$$s = L(x) = \ln(\frac{x - x_0}{x_t - x}) \tag{11}$$

$$x = L^{-1}(s) = \frac{x_t e^s + x_0}{e^s + 1}, \tag{12}$$

where x_0 and x_t are the boundaries of the integration region. Then the PDF of S, $F_S(s)$, is estimated and transformed back to the PDF of X, $F_X(x)$. These two PDFs are related by

$$|F_X(x)dx| = |F_S(s)ds|. \tag{13}$$

Since $L(x)$ is monotonically increasing, the $|...|$ can be dropped and the above equation can be rewritten as

$$F_X(x) = \frac{F_S(s)}{dL^{-1}(s)/ds}, \text{ where} \tag{14}$$

$$F_S(s) = \frac{1}{Nb}\sum_{i=1}^{N} K\left(\frac{L(x) - L(x_i)}{b}\right). \tag{15}$$

Therefore, a better PDF for IS can be obtained. However, there is still a problem because $F_X(x)$ rises to ∞ near the boundaries; simulated annealing does not work for $F_X(x)$. Neither can the PDF be truncated at a certain height because truncation reduces the total probability.

To fix the problem, a n-th degree polynomial PDF, $p(x)$, is used to fit the points sampled evenly from $F_X(x)$. The best-fitted PDF is obtained by minimising the chi-square, which is expressed as

$$\chi^2 = \sum_{i=1}^{N}\left(\frac{p(x) - F_X(x_i)}{F_X(x_i)}\right)^2 \tag{16}$$

$$= \sum_{i=1}^{N}\left(\frac{A_n x^n + A_{n-1}x^{n-1} + ... + A_1 x + C - F_X(x_i)}{F_X(x_i)}\right)^2. \tag{17}$$

The coefficients $(A_1, A_2, ..., A_n)$ can be calculated by solving the linear system:

$$\begin{cases} d\chi^2/dA_1 = 0 \\ d\chi^2/dA_2 = 0 \\ ... \\ d\chi^2/dA_n = 0. \end{cases} \tag{18}$$

The constant term C is not a free variable and it is constructed to make $p(x)$ satisfy $\int_{x_0}^{x_t} p(x) = 1$. Therefore, there are n free variables with n equations, which makes the system solvable.

Then the system is solved using Gaussian elimination. This process (KDE and fitting) can be repeated for m time, where m is defined by the user. In each time, some sample points are picked uniformly from $f(x)$ and the variance of the values of $f(x_i)/p(x_i)$ is computed. At the first time, the coefficients and the variance are recorded in a text file. If the variance becomes smaller in later trials, newly obtained coefficients and variance would override the stored ones. After $p(x)$ finishes being updated, the program reads the coefficients in the file, constructs the PDF, and evaluates the integral using Monte Carlo method with IS.

Using this method, the resultant $p(x)$ closely mimics $F(x)$, and the more times the program is executed, the better $p(x)$ will become.

3.6 Monte Carlo with Adaptive Importance Sampling

The adaptive importance sampling (AIS) is implemented using VEGAS algorithm, which in the beginning uses uniform strata to pick samples, then refines the widths of the strata with the information of the previous samples.

The integration region is firstly divided into M strata, each of which occupies the same probability. Then N samples are drawn from each of them. Denote the i-th sample from stratum j as x_i^j and the height of stratum j as p^j, the estimated area of $f(x)$ in the j-th stratum, denoted as I^j, is computed by $\sum_{i=1}^{N} \frac{f(x_i^j)}{Np^j}$.

Since the estimated areas in the strata are known, every stratum (stratum j) is further divided into M^j sub-divisions, where M^j is proportional to I^j so that the areas of all sub-divisions are approximately the same [3]. After that, $\lfloor \sum_j M^j/M \rfloor$ consecutive sub-sections are regrouped to form a new stratum and the residual sub-divisions are evenly allocated to the last several strata. In this way, the new strata peak at locations where $f(x)$ has higher values, so it is closer to the shape of $F(x)$. Then another set of samples can be extracted using the new strata, and the information of these samples is again used to update the strata. In this way, the PDF converges to $F(x)$.

The PDF cannot approach $F(x)$ indefinitely due to the finite number of strata, so a better implementation would be increasing the number of strata for every update. Then the PDF can be very close to $F(x)$ after several updates.

4 Errors, Results and Discussion

The above methods are used to calculate the integral of a Gaussian function between $x = 0$ and $x = 2$:

$$I = \int_0^2 f(x)dx = \int_0^2 \frac{1}{\sqrt{\pi}} e^{-x^2} dx. \tag{19}$$

The results are analysed below.

4.1 Trapezoidal rule and Simpson's rule

The results computed using the trapezoidal rule and Simpson's rule are shown in Table 1.

Trapezoidal:

Divisions	256	512	1024
$\bar{I}$	0.49766092	0.49766108	0.49766112
ϵ	1.27e-06	3.17e-07	7.92e-08

Simpson:

Divisions	32	64	128
$\bar{I}$	0.49766110	0.49766113	0.49766113
ϵ	1.06e-06	6.60e-08	4.13e-09

Table 1: Estimated Integral $\bar{I}$ and Relative Accuracy ϵ of the Trapezoidal and Simpson's Methods. The method for Simpson's rule converges to the exact value considerably faster that for the trapezoidal rule.

It can be seen that the method that applies Simpson's rule is more accurate than the trapezoidal rule; Simpson's method reaches $\epsilon \approx 1e-6$ with only 32 divisions (5 loops), while the trapezoidal method requires more than 256 divisions to do so. The first eight decimals of the results obtained from Simpson's method stabilised when the number of divisions goes beyond 64, which suggest that the first eight decimals of the exact value are achieved.

The answer given by Wolfram Alpha is 0.4976611325..., which agrees with the results from Simpson's method, and the trapezoidal method does indeed approach the exact value.

4.2 Monte Carlo Methods

Both the NMC method and the MC method with IS are discussed in this section. For IS, two PDFs are used to pick samples:

$$p_1(x) = \frac{1}{2}, x \in [0, 2] \tag{20}$$

$$p_2(x) = -0.48x + 0.98, x \in [0, 2]. \tag{21}$$

The estimated I and the number of samples, N, required to reach desired ϵ for the above methods are shown in Table 2.

NMC:

ϵ	$1e-3$	$1e-4$	$1e-5$	$1e-6$
$\bar{I}$	0.497547	0.497621	0.497696	X
N	1.49e6	1.49e8	1.49e10	X

p1(x):

ϵ	$1e-3$	$1e-4$	$1e-5$	$1e-6$
$\bar{I}$	0.497765	0.497695	0.497674	X
N	6.11e5	6.11e7	6.11e9	X

p2(x):

ϵ	$1e-3$	$1e-4$	$1e-5$	$1e-6$
$\bar{I}$	0.497164	0.497630	0.497656	X
N	7.38e4	7.33e6	7.33e8	X

Table 2: Values of $\bar{I}$ and N for Different Values of ϵ in Different Methods. The values of $\bar{I}$ converge to the exact result as ϵ gets smaller, but the number of samples increases quadratically. And NMC requires the most samples to achieve the same ϵ, while IS with $p_2(x)$ needs the least. The X symbol means that the value cannot be calculated shortly due to the computing power of the personal computer.

The table shows that the sampling PDF has a great impact on N given the same ϵ. $p_2(x)$ has a shape similar to $f(x)$, so it needs the smallest N compared to the other two. Moreover, for each method, as ϵ is scaled by 0.1, N is scaled by 100, which suggests that $\epsilon \propto \frac{1}{\sqrt{N}}$ and it is obvious in Equation 8.

The relation of N and ϵ can be plotted on a log-log coordinates in Figure 4. Since the plot is not stabilised when N is small (Figure 5) and the region with small N is not of interest, only the data with $N > 2e6$ are plotted.

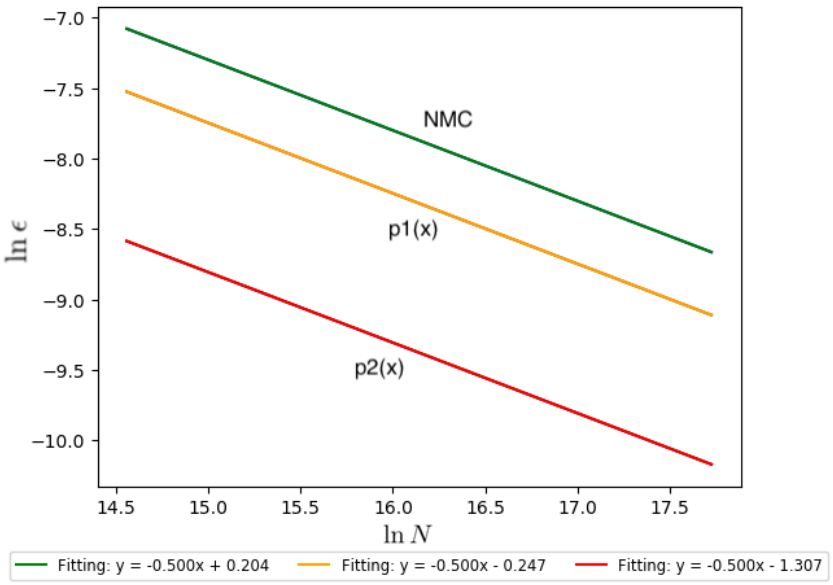
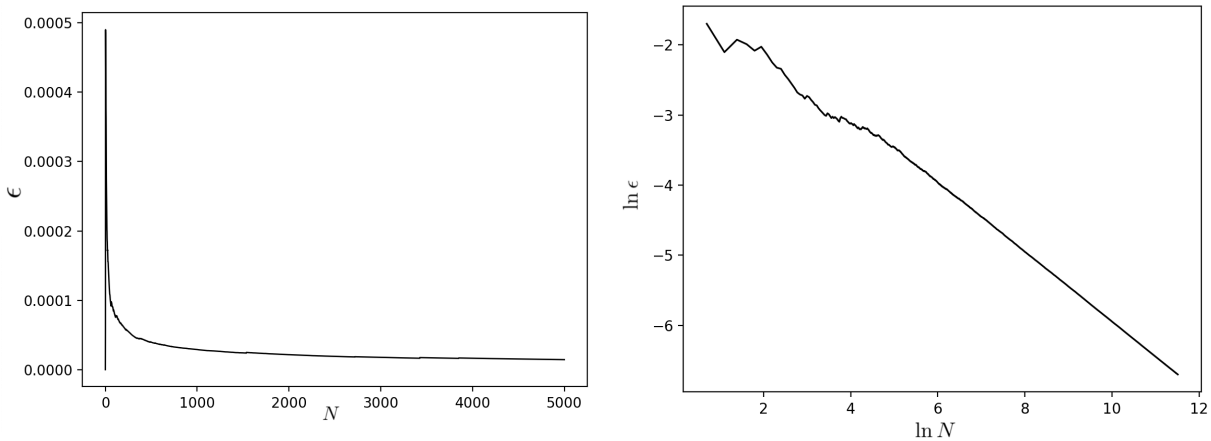


Figure 4: Relation between ϵ and N for Different MC Methods. The data are fitted by linear functions. The functions fit the data so well that they completely cover the original data.



(a) ϵ v.s. N

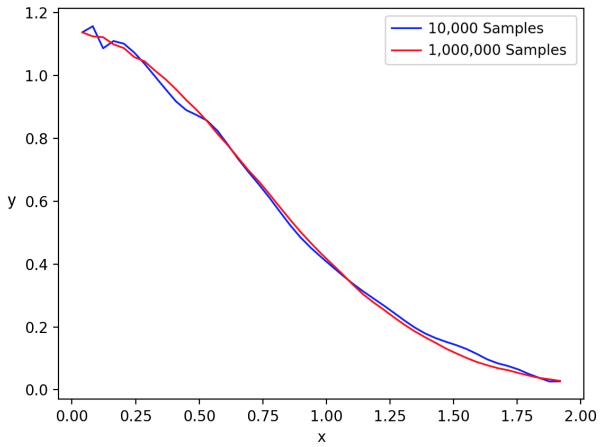
(b) $\ln \epsilon$ v.s. $\ln N$

Figure 5: ϵ in the Region with Small N. ϵ is not stable when N is small, which could introduces extra uncertainty in the relation between ϵ and N, so the data in this region is not plotted in Figure 4

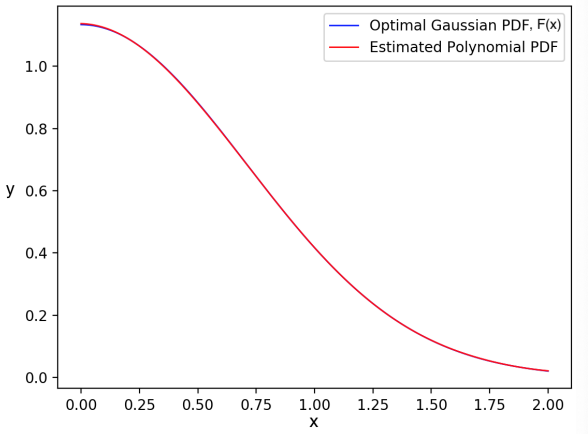
The slopes of the fitting functions in Figure 4 are exclusively -0.500, which agrees the hypothesis that $\epsilon \propto \frac{1}{\sqrt{N}}$. The constant term, C, determines the coefficient of convergence rate: the method with smaller C converges faster. The relation can also be evaluated using Equation 8. In Equation 8, $\langle I_i \rangle^2$ is equivalent to $\int_0^2 f^2(x)/p(x)dx$, so it can be computed with the trapezoidal or Simpson's rule. Then the same relation functions can be derived by reorganising Equation 8.

Using the discovered relation functions, it can be shown that Ns for $\epsilon = 1e-6$ are $1.50e12$, $6.10e11$ and $7.32e10$ for NMC, $p_1(x)$ and $p_2(x)$ respectively.

The method mentioned in Section 3.5 is able to generate KDEs shown in Figure 6a, and a KDE is fitted by a 6-th degree polynomial function, $p(x)$, shown in Figure 6b



(a) KDEs with Different Sample Size. The points at the boundaries are excluded from the plot.



(b) Ploynomial PDF and Optimal Gaussian PDF. The two functions are so close to each other that the Gaussian function is almost invisible

Figure 6: KDEs and the Polynomial PDF. The KDE with a larger sample size produces a smoother PDF. The polynomial function is then used to fit the KDE. The resultant fitting function is a much better PDF for IS.

The polynomial PDF is surely a better choice for IS. A plot of $p(x)/F(x)$ (Figure 7) shows that the percentage difference is much less than 1% in most of the region. Although the plot has higher values near $x = 2$, the value of $F(x)$ is very small there, so the sample size in that region must be small, which does not have a significant impact on the convergence rate.

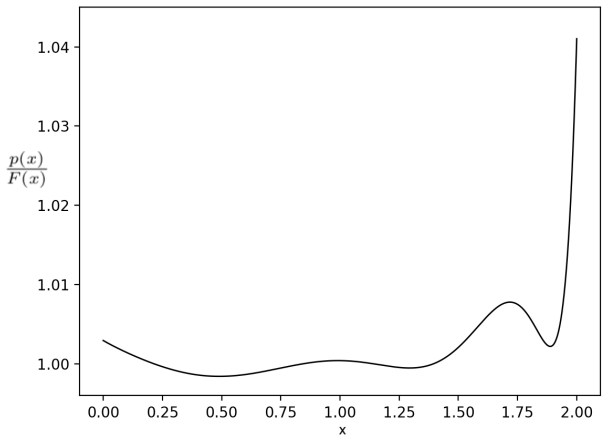
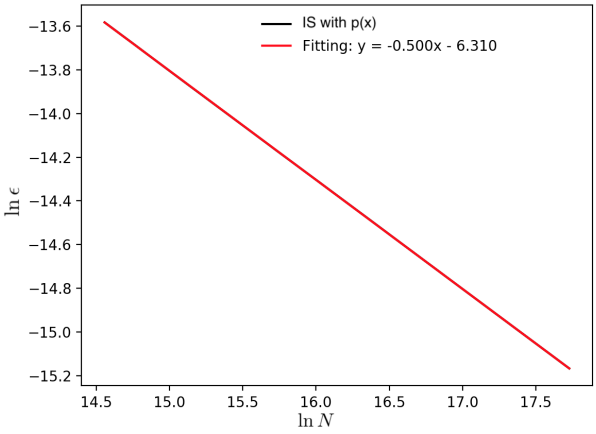
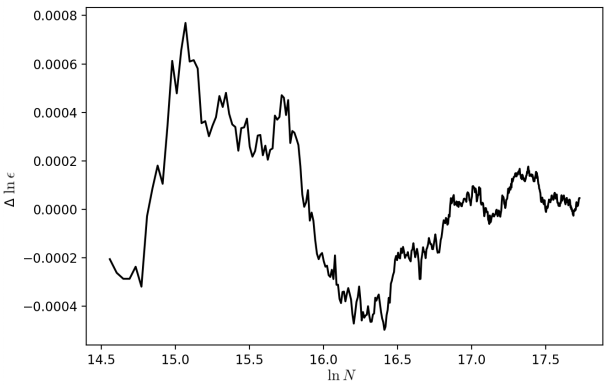


Figure 7: Plot of $p(x)/F(x)$. The difference between p(x) and F(x) is less than 1% within $[0, 1.9]$. Although the difference increases in $[1.9, 2]$, not many samples are drawn from that region (see Figure 6b).

The relation between ϵ and N is also plotted and fitted as shown in Figure 8.



(a) $\ln \epsilon$ v.s. $\ln N$



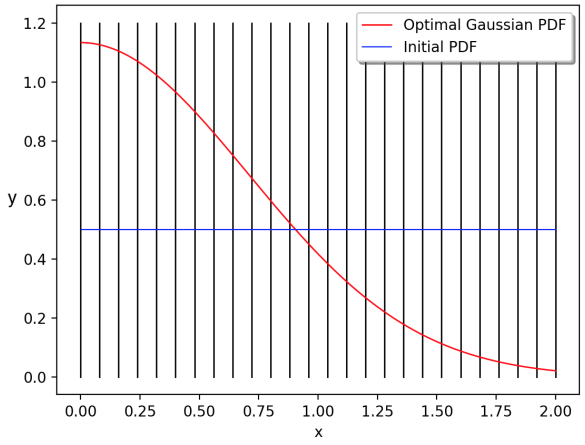
(b) Difference between The Fitting Function and the Data Points.

Figure 8: Fitting Function for $\epsilon - N$ Relation and a Demonstration of the Goodness of the Fitting. The difference between the data points and the function is less than $1e-3$, so the original data points are covered by the fitting function. Note that the difference approaches 0 as N becomes very large.

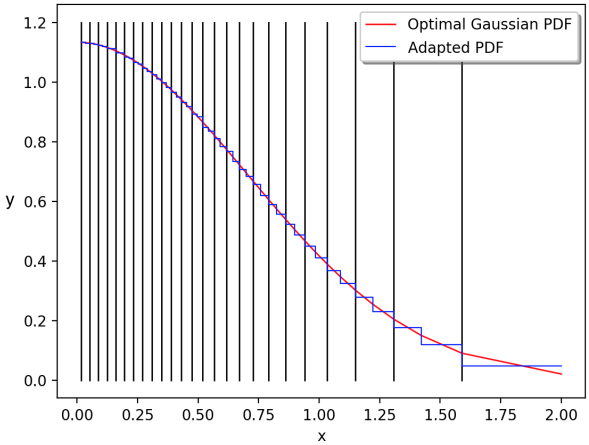
The constant term in the fitting function is -6.310, which suggests a much faster convergence rate than the previous ones. With this polynomial PDF, the result can achieve $\epsilon = 1e-7$ with only $3.32e8$ samples.

4.3 Adaptive Importance Sampling (VEGAS Algorithm)

VEGAS algorithm is demonstrated in Figure 9. The bins automatically approaches the shapes of $F(x)$ as expected, therefore reducing the magnitude of ϵ.



(a) Initial PDF

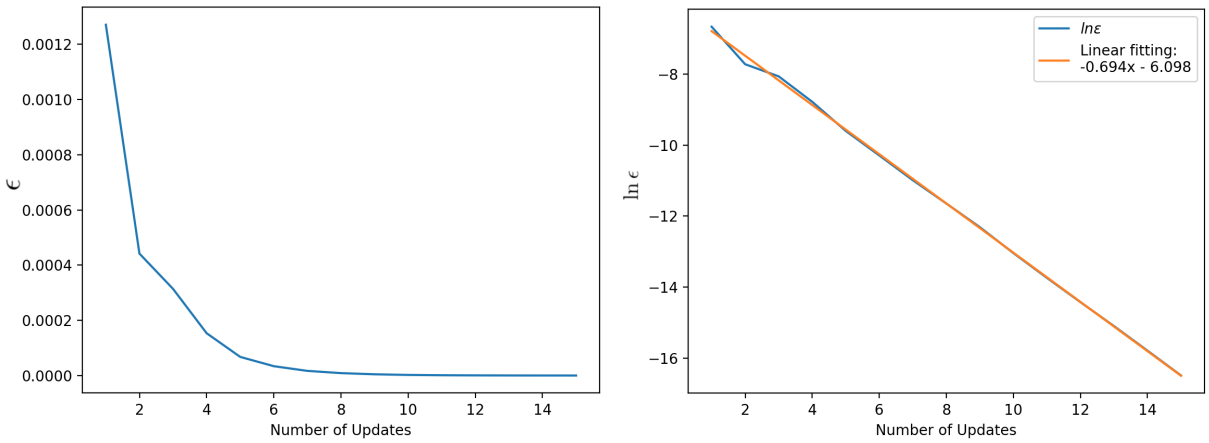


(b) Adapted PDF

Figure 9: Adaptive Process of VEGAS Algorithm. The widths of the bins are refined using the information of previously sampled points so that the PDF becomes closer to the optimal one. The black lines show how the bins are distributed.

Using this method, ϵ decreases exponentially after each cycle. The plot of ϵ versus the number of updates is illustrated in Figure 10. The program does 12 updates to achieve $\epsilon = 1e-6$ and 15 updates to achieve $\epsilon = 1e-7$, and the calculation can be done less than 10 seconds.

The plot of $\ln \epsilon$ as a function of the number of updates, N', is fitted by a straight line. The resultant fitting function suggests that $\epsilon \propto e^{-0.694N'}$.



(a) Plot of ϵ v.s. Number of Updates

(b) Plot of $\ln \epsilon$ v.s. Number of Updates

Figure 10: Relation of ϵ and Number of Updates in VEGAS Algorithm. ϵ decreases exponentially with the number of updates. The result achieves $\epsilon = 1e-7$ at the 15-th update

5 Conclusion

In this project, different numerical integration methods were implemented. The MC methods converge slowly in 1D cases as $\epsilon \propto \frac{1}{\sqrt{N}}$. VEGAS algorithm and the methods implemented for trapezoidal and Simpson's rules all achieve exponential convergence; therefore, to integrate the PDF of a 1D quantum system, it is optimal to choose one of these methods instead of MC methods. The project could be improved by extending to multi-dimensional systems, in which MC methods may possibly have advantages over Newton-Coates methods.

References

[1] Giles, M. (2018). Numerical Methods II. [online] People.maths.ox.ac.uk. Available at: https://people.maths.ox.ac.uk/gilesm/mc/mc/lec4.pdf [Accessed 16 Dec. 2018].

[2] Silverman, B. (2018). Density Estimation for Statistics and Data Analysis. Boca Raton: Routledge, p.48.

[3] Chrzaszcz, M. (2018). Adaptive Monte CarloIntegration Methods. [online] Available at: https://www.uzh.ch/cmsssl/physik/dam/jcr:ff1992bc-e15f-4602-9c61-268b8f1d5041/lec3.pdf [Accessed 16 Dec. 2018].