Investigation of One-dimensional Numerical Integrators for Quantum Systems

Muchen Li
Blackett Laboratory, Imperial College London
(Dated: 16 December 2019)

We investigate the performance of five one-dimensional numerical integrators, characterised by the number of integrand evaluations required, n, to achieve a given error, using a Gaussian test integrand. We demonstrate that the extended trapezoidal rule has $O(n^{-2})$, the extended Simpson's rule $O(n^{-4})$, and Monte Carlo (MC) integration techniques $O(n^{-1/2})$ global error. Consequently, we find, to reach a relative error of 10^{-6}, $n = 513;\ 65$ for the extended trapezoidal and Simpson's rule respectively. The same error should be achieved if a basic MC integrator was run for ~ 35 days, with $n \approx 6.1 \times 10^{11}$, or if a linear importance sampling MC integrator with optimised parameters was run for ~ 6 days, with $n \approx 7.4 \times 10^{10}$. Furthermore, we show that the VEGAS algorithm, an adaptive MC integrator, performs much better than the earlier two MC methods, taking approximately 3 seconds to run, with $n \approx 3.4 \times 10^5$, provided its input parameters are optimally chosen.

I. Introduction

Quantum mechanics remains one of our most versatile and robust physical theories today. Its widespread use, ranging from the realms of particle physics, to astrophysics, is certainly evidence for its success. Yet, the most primitive equation in the theory, the time independent Schrödinger equation:

$$\hat{H}\psi_n = E_n\psi_n \ , \tag{1.1}$$

where ψ_n are the energy eigenstates, E_n the associated energy eigenvalues, and $\hat{H}$ the Hamiltonian operator, remains analytically soluble only in the simplest circumstances. It is clear that numerical techniques are required to analyse real systems with many particles and complicated potentials.

Suppose we obtain one-dimensional numerical solutions for ψ_n and E_n by numerically solving the eigenvalue equation (1.1) in position representation, for a particular particle. An important quantity to now find is the probability the particle lies between a and b:

$$P(a \leq x < b) = \int_a^b |\psi_n(x)|^2 \, \mathrm{d}x \ . \tag{1.2}$$

Since the typical techniques used to solve the TISE are finite difference methods, we would obtain samples of ψ_n, rather than an analytical expression. It is therefore clear that a numerical integrator is required. In this project, we present several numerical integrators and analyse them in detail using Python. We firstly evaluate the order of five one-dimensional numerical integrators. Two integrators are from the class of Newton–Cotes rules, and three are Monte Carlo methods. The latter would additionally require the use of an interpolator to obtain an expression for ψ_n from its samples.

The performance of each method are then compared, where the performance is characterised by the number of integrand evaluations required to reach a specified (low) relative error. A Gaussian integrand is used throughout. We discuss why two Monte Carlo methods cannot converge sufficiently quickly, and give an estimate of the time it should take for convergence. Finally, we briefly mention expected findings in higher dimensions, and explain the true power of Monte Carlo integrators.

Besides simply computing probabilities, numerical integrators can be used for other purposes in a quantum system. An example would be finding expectation values such as $\langle x \rangle$ or $\langle p \rangle$ for well-localised particles: the latter would also require estimating a derivative by finite difference techniques. Although seemingly trivial, refinements of numerical integrators are still an ongoing subject of research [1], as their use are frequent and widespread: quantum mechanics is certainly not the only discipline that finds a use for them.

II. Theory and Method

We concern ourselves with the (normalised) free particle Gaussian wave packet, in position representation [2]:

$$\psi(x) = \frac{1}{(2\pi {\sigma_x}^2)^{1/4}} \exp\left[-\left(\frac{x-\mu_x}{2{\sigma_x}^2}\right)^2\right] \exp\left(\frac{i\mu_p x}{\hbar}\right), \tag{2.1}$$

where μ_x is the position mean, ${\sigma_x}^2$ the position variance, and μ_p the momentum mean. The Gaussian profile reflects that infinitely localised position or momentum states cannot exist in reality, due to the finite size, and thus precision, of measuring apparatuses. The phase factor $\exp\left(\frac{i\mu_p x}{\hbar}\right)$ ensures the wave packet's momentum representation also has a Gaussian profile.

We choose $\mu_x = 0$, and ${\sigma_x}^2 = 1/2$. Now,

$$\psi(x) = \frac{1}{\pi^{1/4}} \exp\left(-\frac{x^2}{2}\right) \exp\left(\frac{i\mu_p x}{\hbar}\right) \ , \tag{2.2}$$

such that its modulus squared is

$$|\psi(x)|^2 = \frac{1}{\sqrt{\pi}} \exp(-x^2) \ , \tag{2.3}$$

and its associated probability $P(a \leq x < b)$ with $a = 0$, and $b = 2$, is

$$\frac{\mathrm{erf}(2)}{2} \approx 0.49766113 \text{ (to 8 s.f.) } , \tag{2.4}$$

as expected, as this is the area of a Gaussian from the centre to ~3 sigma's away. Note the error function is defined as [3]

$$\mathrm{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, \mathrm{d}t \ . \tag{2.5}$$

Equation (2.3) will be used as the test function $f(x)$ that will generate the integrand samples for our performance comparison of numerical integration methods.

A. Newton–Cotes Rules

Newton–Cotes rules are methods that divide an integrand into a number of sub-regions with a constant step size, and for each sub-region, interpolates its integrand data points by some Lagrange polynomial. The resulting integral of the Lagrange polynomials are simply weighted sums of the integrand data points. We are interested in *closed* Newton–Cotes rules, where the integrand is directly evaluated at the limits a and b.

The simplest closed Newton–Cotes rule is the trapezoidal rule. An integrand $f(x)$ is divided into n equal regions of step size $h \sim n^{-1}$, and is sampled at the $(n+1)$ points. Given two adjacent data points $f(x_i)$ and $f(x_{i+1}) = f(x_i + h)$, a straight line is used as the interpolant. The resulting shape is a trapezium, with area

$$\int_{x_i}^{x_{i+1}} f(x)\, \mathrm{d}x = \frac{h}{2}[f(x_i) + f(x_{i+1})] \ . \tag{2.6}$$

This is the trapezoidal rule. Patching together all n regions yields

$$\int_a^b f(x)\, \mathrm{d}x = \frac{h}{2}[f(a) + f(b) + 2f(a+h) + \cdots + 2f(b-h)] \ . \tag{2.7}$$

This is the *extended* trapezoidal rule. It is shown graphically in Fig. 1. The trapezoidal rule has an error term of order h^3, so the extended trapezoidal rule has a global error of order $nh^3 \sim h^2$.

In order to compare performance between methods, we need to iterate the extended trapezoidal rule until a desired error is reached [4]. This can be achieved by firstly evaluating

$$T_0 = \frac{h_0}{2}[f(a) + f(b)] \ , \tag{2.8}$$

with $h_0 = b - a$. This is the extended trapezoidal rule with $n = 1$. Successive iterations T_k double the previous iteration's n, and uses T_{k-1} such that f is never evaluated at the same point twice:

$$T_k = \frac{T_{k-1}}{2} + h_k \sum_{i=1}^{2^{k-1}} f[a + (2i-1)h_k] \ , \tag{2.9}$$

with $h_k = h_{k-1}/2 = 2^{-k}(b-a)$. The relative error convergence condition is then

$$\left|\frac{T_k - T_{k-1}}{T_{k-1}}\right| < \epsilon \ , \tag{2.10}$$

for some chosen relative error threshold ϵ. The first T_k is then the outputted estimate of the integral. This method is superior to the basic extended trapezoidal rule (2.7), repeatedly iterated for different h, as the fact that f is never evaluated at the same point twice means the computation time is significantly reduced.

The next closed Newton–Cotes rule is Simpson's rule, which interpolates every three neighbouring points with a quadratic Lagrange polynomial:

$$\int_{x_i}^{x_{i+2}} f(x)\, \mathrm{d}x = \frac{h}{3}[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})] \ . \tag{2.11}$$

Patching together all n regions yields

$$\int_a^b f(x)\, \mathrm{d}x = \frac{h}{3}[f(a) + f(b) + 4f(a+h) + \cdots + 4f(b-h) + 2f(a+2h) + \cdots + 2f(b-2h)] \ , \tag{2.12}$$

which is the *extended* Simpson's rule. It is shown graphically in Fig. 1. The global error is of order h^4.

An iterative form of the extended Simpson's rule is [4]:

$$S_k = \frac{4}{3} T_{k+1} - \frac{1}{3} T_k \ , \tag{2.13}$$

where S_k is the kth iteration of the extended Simpson's rule, corresponding to a step size $h_k = 2^{-(k+1)}(b-a)$. The extra factor of 2^{-1} is due to the requirement that $h_0 = 2^{-1}(b-a)$, as Simpson's rule requires three data points, rather than two as in the trapezoidal rule.

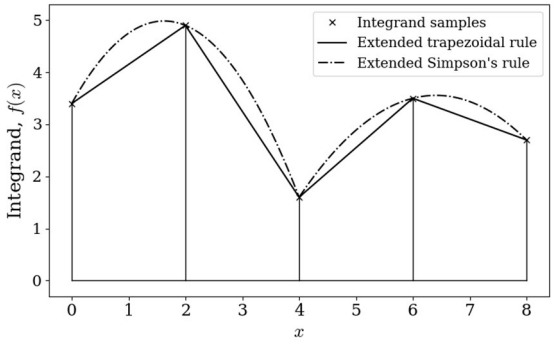

FIG. 1. A graphical illustration of the extended trapezoidal rule and extended Simpson's rule, with $n = 4$ regions. The data points are samples of the arbitrarily chosen integrand $f(x)$. The constant sampling step size is $h = 2$. A definite integral of $f(x)$ is approximated by interpolating between the data points and evaluating the resulting area. The extended trapezoidal rule uses linear interpolation, with an area given by (2.7). The extended Simpson's rule uses Lagrange quadratics between every three neighbouring points, with an area given by (2.12).

This means an implementation of the iterative extended Simpson's rule only requires an appropriate sum of (2.8) and (2.9) with the correct k. Its convergence condition is

$$\left|\frac{S_k - S_{k-1}}{S_{k-1}}\right| < \epsilon \ . \tag{2.14}$$

B. Monte Carlo Techniques

Consider again the definite integral of $f(x)$. This can be rewritten as

$$I = \int_a^b \frac{f(x)}{\rho(x)}\rho(x)\,\mathrm{d}x = \int_a^b g(x)\,\rho(x)\,\mathrm{d}x = \langle g\rangle \ , \tag{2.15}$$

where $g(x) \equiv \frac{f(x)}{\rho(x)}$, and $\rho(x)$ is a chosen probability density function (PDF), with $x \in [a, b)$. We can therefore estimate I by sampling $g(x)$ for a total of n times, and evaluating

$$\hat{I} = \widehat{\langle g\rangle} = \frac{1}{n}\sum_{i=1}^{n} g(x_i) \ , \tag{2.16}$$

where the x_i are distributed according to $\rho(x)$. The sample variance of the $g_i = g(x_i)$ is

$${s_{g_i}}^2 = \frac{n}{n-1}\left(\widehat{\langle g^2\rangle} - \widehat{\langle g\rangle}^2\right) \ . \tag{2.17}$$

The error of $\hat{I}$ is therefore

$$s_{\hat{I}} = \frac{s_{g_i}}{\sqrt{n}} \ . \tag{2.18}$$

Note that since s_{g_i} is an estimator of σ_{g_i}, the true variance of the g_i, and that the latter does not scale with n, then this should also be true of s_{g_i}. These equations are known as Monte Carlo (MC) integration methods; all have errors of $O(n^{-1/2})$ by (2.18). It is easy to turn these into iterative methods, where we repeatedly increase n until the relative error convergence condition

$$\left|\frac{s_{\hat{I}}}{\hat{I}}\right| < \epsilon \tag{2.19}$$

is reached.

We now consider choices of $\rho(x)$. The simplest is the uniform deviate $\rho(x) = \frac{1}{b-a}$. This is the basic MC integration method, and it can be shown that $\hat{I} = (b-a)\widehat{\langle f\rangle}$, ${s_{f_i}}^2 = \frac{n}{n-1}\left(\widehat{\langle f^2\rangle} - \widehat{\langle f\rangle}^2\right)$, and $s_{\hat{I}} = \frac{b-a}{\sqrt{n}} s_{f_i}$ [4]. We use Python's built-in Mersenne Twister algorithm as the uniform deviate.

Faster convergence can be achieved by carefully choosing a $\rho(x)$ such that $\sigma_{g_i} < \sigma_{f_i}$. This is possible if the range of $g(x)$ is narrower than that of $f(x)$. Therefore, suitable choices of $\rho(x)$ involve finding functions that closely match the shape of $f(x)$, then normalising it to become a PDF. This technique is known as importance sampling, and a well chosen $\rho(x)$ will cause the algorithm to converge faster by having a decreased s_{g_i}, compared to the basic MC method.

An obvious choice is $\rho(x) = k|f(x)|$. Indeed, this converges after only one iteration [5]. However, the constant of proportionality can be found as $k = {I_{|f(x)|}}^{-1}$. In other words, we need to know I to evaluate itself, so this choice is unsuitable. A more useful choice is to generalise $\rho(x)$ to polynomials with well-chosen coefficients. We consider a first order polynomial:

$$\rho(x) = Ax + B \ , \tag{2.20}$$

where A, B are specifically chosen for every $f(x)$. Plots of two integrands, (2.20), and the resulting $g(x)$, are shown in Figs. 2 and 3. We immediately see from the plots, that $\sigma_{g_i} < \sigma_{f_i}$, so faster convergence are expected.

We use the transformational method on the $y \in [0, 1)$ uniform deviate to generate x_i according to (2.20). It is shown in the Appendix that this transformation is:

$$x_\pm = -\frac{B}{A} \pm \sqrt{\left(\frac{B}{A} + a\right)^2 + \frac{2y}{A}} \ , \tag{2.21}$$

where the positive transformation is taken for

$$\frac{B}{A} + a \geq 0 \ , \tag{2.22}$$

and the negative transformation otherwise.

Better $\rho(x)$ can be chosen by including higher order polynomial terms. This requires fixing higher degrees of freedom arising from the coefficients, and more involved

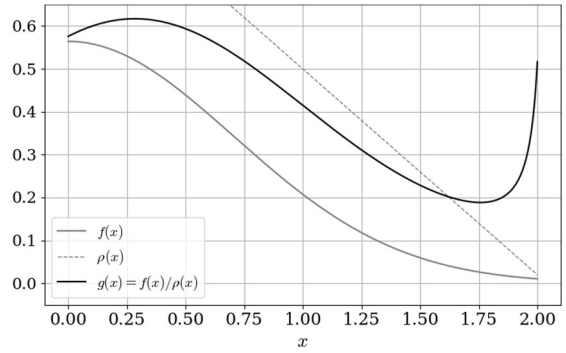

FIG. 2. A plot of the modulus squared of the Gaussian wavefunction $f(x) = |\psi(x)|^2$, a linear importance distribution $\rho(x) = Ax + B$ with $A = -0.48$; $B = 0.98$, and their ratio $g(x)$. The importance distribution $\rho(x)$ is correctly normalised. Note an additional requirement that $\rho(x)$ cannot be zero for $x \in [a, b)$, or $g(x)$ will diverge, which is satisfied for these A and B. We see that the range of $g(x)$ is smaller than that of $f(x)$ close to the origin, and a sample of x_i drawn from $\rho(x)$ are more likely to be located near the origin. Therefore, we see $\sigma_{g_i} < \sigma_{f_i}$ and expect a faster convergence time when using MC integration.

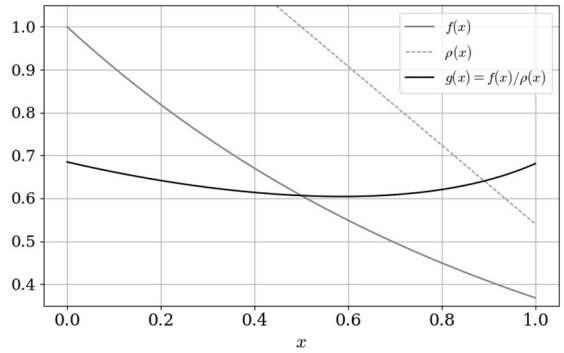

FIG. 3. A plot of a decaying exponential $f(x) = e^{-x}$, a linear importance distribution $\rho(x) = Ax + B$ with $A = -0.92$; $B = 1.46$, and their ratio $g(x)$. Again, we see $\sigma_{g_i} < \sigma_{f_i}$, and expect faster MC convergence as a result.

transformations. However, in general, manual selection of good $\rho(x)$ functions is time-consuming, and detailed knowledge about $f(x)$ is required. The class of *adaptive* importance sampling MC integrators solves this issue by using samples of $f(x)$ to adapt an initially simple $\rho(x)$ into the form ${I_{|f(x)|}}^{-1}|f(x)|$ as best as possible. We investigate the VEGAS algorithm (without damping) here [6].

The algorithm is split into two phases. The first phase begins with a uniform distribution $\rho(x)$, split into N equal step functions, each with their areas preserved throughout. In each iteration, we take M samples of $f(x)$ that lie within the ith step function with width Δx_i, then we evaluate

$$\bar{f}_i \equiv \sum_{j=1}^{M} \left|f(x_j)\right| \ , \tag{2.23}$$

and

$$m_i = K \frac{\bar{f}_i \Delta x_i}{\sum_i \bar{f}_i \Delta x_i} \ , \tag{2.24}$$

where $K > 0$ is a chosen factor. Each step function is further subdivided into $m_i + 1$ subregions. Note, by construction, the total number of subregions is $K + N$. Then, all neighbouring

$$\frac{K+N}{N} \tag{2.25}$$

subregion *widths* are amalgamated into N new step functions. This procedure is iterated C times.

Since m_i is proportional to the area of $f(x)$ within the step function, the net result of an iteration is that the steps adjust themselves to closer match the area of $f(x)$, until where all m_i becomes approximately equal. This is demonstrated graphically in Fig. 4. The integral estimate take no contributions from the first phase, since σ_{g_i} is relatively large as the optimal $\rho(x)$ is being found [5].

The second phase is a standard MC integration routine with importance sampling, with $\rho(x)$ given by the result of the first phase. Importantly, it is easy to transform a uniform deviate into a sum of step functions with equal area. This is what makes this phase particularly simple to implement, and fast to execute.

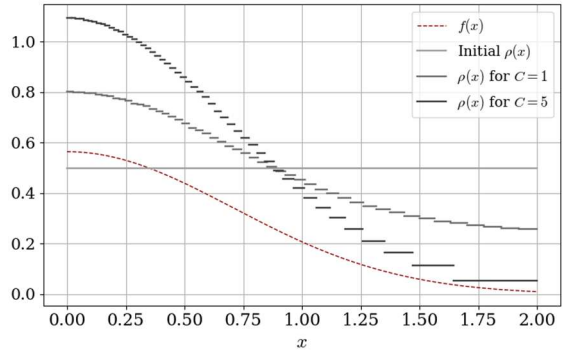

FIG. 4. A plot of how $\rho(x)$ evolves as the number of refinement iterations increases for the first phase of the adaptive importance sampling MC integrator. The integrand is $f(x) = |\psi(x)|^2$, and the chosen parameters are $N = 50$; $M = 3$; $K = 50$. Note that in reality, for this $f(x)$, K should be at least $5N$ to ensure fast convergence [6]. We choose $K = 50$ here illustratively to force the first iteration to converge slower. It can be seen that the iterations gradually refine an initially uniform distribution into step functions that together resemble a rescaled $f(x)$ with unitary area. This is due to the subdivision formula (2.24) that cause more subdivisions in step functions where $f(x)$ is high for constant widths. As a result, after an amalgamation, their widths will be reduced, and since area is preserved, their heights increased. We expect a particularly low σ_{g_i} for $C = 5$.

III. Testing of Methods

We used three test integrands: a quadratic, a decaying exponential, and a sinusoid to test the integrators. They were chosen because their integrals are well known, and each have a different range and behaviour. All

integrators were used on these, and it was checked that their outputs matched the exact result to approximately the specified relative error. The error was chosen to be small (10^{-3} or 10^{-6}) such that the number of integrand evaluations were also checked to have followed the expected ascending order: Simpson's–trapezoidal–adaptive MC–linear importance MC–basic MC (see Section IV). The linear importance sampling MC method was not used on the sinusoid, as a linear $\rho(x)$ is clearly unsuitable for an oscillatory function.

Specific to the two Newton–Cotes rules, we checked that the number of integrand evaluations, $(n+1)$, was related to k_{final} by

$$\begin{aligned}(n_T+1) &= 2+1+2+4+\cdots+2^{k_{\text{final}}-1} \\ &= 2^{k_{\text{final}}}+1 \ , \end{aligned} \tag{3.1}$$

for the extended trapezoidal rule, and

$$\begin{aligned}(n_S+1) &= 3+2+4+8+\cdots+2^{k_{\text{final}}} \\ &= 2^{k_{\text{final}}+1}+1 \ , \end{aligned} \tag{3.2}$$

for the extended Simpson's rule.

Additionally, for the MC integrators, we checked that the s_{f_i} or s_{g_i} for the different methods were in the expected ascending order: adaptive MC–linear importance MC–basic MC. We also checked that the implemented transformations from a uniform deviate to the desired PDFs gave correct distributions of numbers. Finally, the adaptive MC method was checked to produce step functions closely resembling $|f(x)|$ after sufficient refinements.

IV. Results and Discussion

We assume throughout, that integrand evaluations are far more computationally expensive than other operations, such as comparison statements, and therefore the number of required integrand evaluations to reach a specified relative error threshold acts as a direct measure of the performance of each method.

A. Newton–Cotes Rules

The implementation of the Newton–Cotes rules gave integral estimates, with $\epsilon = 10^{-6}$,

$$\begin{aligned}\hat{I}_T &= 0.49766108 \text{ (to 8 s.f.) ;} \\ \hat{I}_S &= 0.49766113 \text{ (to 8 s.f.) ,}\end{aligned} \tag{4.1}$$

for the extended trapezoidal and Simpson's rules, respectively. These agree well with (2.4) within ϵ. The number of integrand evaluations were

$$\begin{aligned}(n_T+1) &= 513 \ ; \\ (n_S+1) &= 65 \ , \end{aligned} \tag{4.2}$$

and both methods took a negligible amount of time to run. To analyse these, consider again the trapezoidal convergence criteria (2.10). We have $T_k = I + s_T$, where I is the exact integral, and $s_T \sim O({h_k}^2)$ is the final error. Noting that $h_k = h_{k-1}/2$, and the denominator $\sim |I|$, we can rewrite (2.10) as

$$s_T \sim \frac{|I|\epsilon}{3} \ . \tag{4.3}$$

Since $h_k \sim n^{-1}$, we have an estimate for n_T:

$$n_T \sim \left(\frac{3}{|I|\epsilon}\right)^{\frac{1}{2}} \sim 2500 \ . \tag{4.4}$$

Likewise, for the extended Simpson's rule,

$$n_S \sim \left(\frac{15}{|I|\epsilon}\right)^{\frac{1}{4}} \sim 74 \ . \tag{4.5}$$

These agree with (4.2) to within an order of magnitude, showing the order of each method is correct. The loss of precision here is due to approximating numerical factors in s to unity. We see that when a very accurate integral is required, these Newton–Cotes rules are very good performing methods, and that Simpson's rule performs better than the trapezoidal rule.

B. Monte Carlo Techniques

Consider now the MC convergence criteria (2.19). Together with (2.18), we have

$$n \sim \left|\frac{s_{g_i}}{\hat{I}}\right|^2 \epsilon^{-2} \ , \tag{4.6}$$

or equivalently,

$$\log(n) \ \sim \ -2\log(\epsilon) + 2\log\left|\frac{s_{g_i}}{\hat{I}}\right| \ . \tag{4.7}$$

Fig. 5 shows a plot of $\log(n)$ against $\log(\epsilon)$ for the basic, and linear importance sampling MC integrators. The gradient of the fitting lines are -2.000 ± 0.003; -2.03 ± 0.01, demonstrating that the MC integrators have $O(n^{-1/2})$ errors. The y-intercept of the fitting lines are -0.213 ± 0.006; -1.22 ± 0.03, corresponding to $s_{f_i} = 0.195 \pm 0.003$; $s_{g_i} = 0.123 \pm 0.004$, for the two methods respectively. This agrees well with the outputted estimates, $s_{f_i} = 0.1946$; $s_{g_i} = 0.1353$, using the smallest error threshold data point $\epsilon = 10^{-3}$, showing n scales with s as expected. In addition, it also shows that the importance sampling method is superior, and that a well chosen $\rho(x)$ can lead to considerably faster convergence.

We did not try any $\epsilon < 10^{-3}$, as $\epsilon_1 = 10^{-3}$ already took considerable time ($t \approx 3$ and ≈ 0.5 seconds respectively) for the routines to return an output. Since $t \propto n \propto \epsilon^{-2}$, it is expected that to reach $\epsilon_2 = 10^{-6}$,

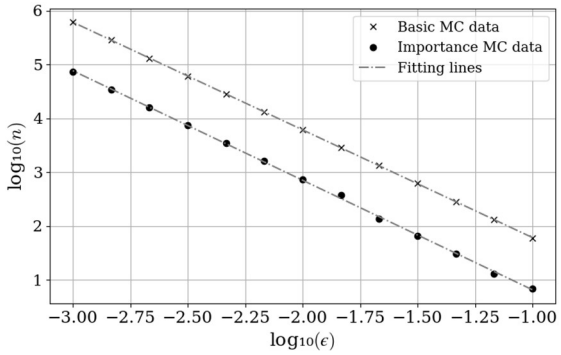

FIG. 5. A plot of n as a function of ϵ in log space. We expect, by (4.7), a straight line passing through either data with the same gradient -2, representing the order of the MC techniques. The y-intercept of the lines are expected to differ, representing that the importance sampling MC technique has lower errors for samples of the (modified) integrand.

$$n_2 \approx n_1 \left(\frac{\epsilon_2}{\epsilon_1}\right)^{-2} \approx 6.1 \times 10^{11} \ , \tag{4.8}$$

where $n_1 = 611897$, for the basic MC routine. This corresponds to a time of approximately 35 days. Similarly, $n_2 \approx 7.4 \times 10^{10}$ for the importance sampling MC routine, with $n_1 = 74121$. The expected time taken is 6 days. This shows in 1D, Monte Carlo methods are extremely weak performing methods, and quickly becomes impractical for use as the required integral accuracy increases.

Lastly, we investigate the VEGAS algorithm by attempting to evaluate $\hat{I}$ with $\epsilon = 10^{-6}$. The goal is to find a set of optimised parameters C, M, N, and K such that $n = n^{(1)} + n^{(2)}$ is minimalised for any chosen ϵ, where $n^{(1)}$ and $n^{(2)}$ are the number of integral evaluations in the first and second phases, respectively. Note that

$$n^{(1)} = CMN \ . \tag{4.9}$$

Firstly, given a fixed N, it is M and K that controls the speed of convergence of the step functions to ${I_{|f(x)|}}^{-1}|f(x)|$. We find that the choice of K affects this minimally provided it is larger than N. We also note that $n^{(1)}$ is independent of K. We therefore use a safety factor of 10, such that $K = 10N$. In addition, since $n^{(1)} \propto M$, M should be made as small as possible provided the step functions converge after only a few iterations. Therefore, we use the choice $M = 3$.

Now, since N controls the number of step functions used for $\rho(x)$, and therefore how well they can approximate the integrand provided $\rho(x)$ converges, we expect only N to directly affect σ_{g_i}, provided convergence of the step functions are reached. A larger choice of N would decrease σ_{g_i}, but simultaneously increase $n^{(1)}$. A balance of the two factors should therefore be reached for some N. If a very accurate integral is required, such as when $\epsilon = 10^{-6}$ here, then any arbitrary initial choice of N is likely to be computationally unfeasible. A better initial choice can be determined by starting close to the optimal N for slightly less accurate integrals.

Lastly, C should be made minimal provided the step functions converge. We included a method in our integrator to let the user be able to determine this. Again, a good initial choice of C starts close to its optimal choice for less accurate integrals. It is also expected that C is close to unity [5].

We find that with $N \sim 10^4$, the VEGAS algorithm is indeed computationally feasible. We therefore plot $\log[n^{(2)}]$ against $\log(N)$ with all other parameters fixed in Fig. 6. Interestingly, a straight line emerges with gradient -2, implying a power law relationship

$$n^{(2)} = 10^c N^{-2} \ , \tag{4.10}$$

where $c = 13.15 \pm 0.13$ is the fitting line y-intercept, that we expect to depend on ϵ in general (likewise for the gradient). Note that this implies $s_{g_i} \propto N^{-1}$ for $\epsilon = 10^{-6}$. Therefore,

$$n = CMN + 10^c N^{-2} \ , \tag{4.11}$$

and this should be minimalised if an optimal N is used. By differentiating and solving, we find

$$N_{\text{op}} = \left(\frac{2 \cdot 10^c}{CM}\right)^{1/3} \approx 11621 \tag{4.12}$$

for $C = 6$. Further differentiation shows this is indeed a minimum in n. Using this optimised value of N, and for $C = 6$ (we have removed the safety factor of 1.5 used in Fig. 6), we compute $\hat{I} = 0.49766090$ (to 8 d. p.), with $n^{(1)} = 209178$; $n^{(2)} = 128391$. The entire routine took $\approx$ 3 seconds to run. The integral estimate agrees with (2.4) within ϵ. This shows the VEGAS algorithm is extremely powerful compared to the other two outlined MC methods.

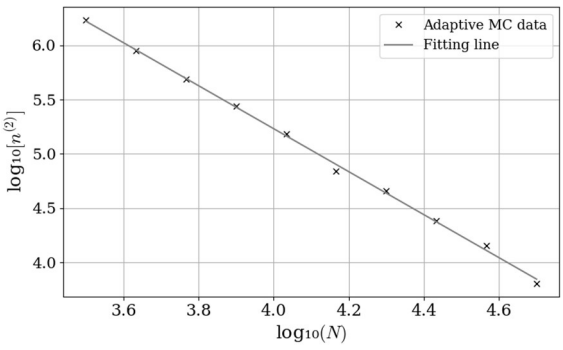

FIG. 6. A plot of $n^{(2)}$ as a function of N in log space, for $\epsilon = 10^{-6}$. We choose $K = 10N$, $M = 3$ as discussed, and $C = 9$ as it was found that for $N = 10^4$, the optimal $C = 6$. This is multiplied by a safety factor 1.5. The fact that these data points can be computed shows achieving this high accuracy is computationally feasible. A straight line emerges with gradient -1.98 ± 0.03 and y-intercept 13.15 ± 0.13, showing a power law relationship.

V. Conclusions

We presented five numerical integrators in 1D, each with a specified order. We have seen that the extended trapezoidal rule has global error of $O(n^{-2})$, and the extended Simpson's rule has global error of $O(n^{-4})$. This makes them particularly good in 1D for evaluating definite integrals, and it was shown that 513 and 65 (respectively) integrand evaluations were required to reach $\epsilon = 10^{-6}$. We have seen that the class of Monte Carlo techniques, with errors of $O(n^{-1/2})$, are inferior to those, and with the exception of VEGAS, an MC method with adaptive importance sampling, the basic and linear importance sampling MC methods failed to provide an integral estimate for $\epsilon = 10^{-6}$. We argued that VEGAS has superior performance, and is more versatile than the other two MC methods presented, and demonstrated it by explicitly evaluating an integral, with ≈ 340000 integrand evaluations, for $\epsilon = 10^{-6}$.

Although we have seen that Newton–Cotes rules are far superior in 1D, it is known that their global errors depend on the dimensionality d of the integral as $O(n^{-2/d})$, and $O(n^{-4/d})$, for the extended trapezoidal and Simpson's rules respectively [4]. Monte Carlo methods however, have errors of $O(n^{-1/2})$ independent of d [4]. We therefore expect MC methods to outperform the two Newton–Cotes rules for $d > 8$. Although single particle quantum systems have wavefunctions of at most $d = 3$, there exist systems with multiple particles where their wavefunctions depend also on the positions of other particles: $\psi_i(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_i, \dots)$. This is where MC techniques shine as the superior class of methods, and where an adaptive MC method, like VEGAS, becomes the method of choice for numerical integration.

Appendix

Consider the PDF given by

$$\rho(x) = Ax + B \ , \tag{A.1}$$

in the interval $x \in [a, b)$, where A and B are related, through normalisation, by

$$\frac{Ab^2}{2} + Bb = 1 + \frac{Aa^2}{2} + Ba \ . \tag{A.2}$$

We wish to obtain the transformation $x(y)$ such that samples y_i drawn from a uniform deviate $[0, 1)$ are mapped to x_i that are distributed according to $\rho(x)$. The cumulative density function of $\rho(x)$ is

$$P(x) = \int_a^x (At + B)\,\mathrm{d}t = \frac{Ax^2}{2} + Bx - \frac{Aa^2}{2} - Ba \ . \tag{A.3}$$

The transformational method states that $x = P^{-1}(y)$. So, the transformation $x(y)$ is

$$x_\pm = -\frac{B}{A} \pm \sqrt{\left(\frac{B}{A} + a\right)^2 + \frac{2y}{A}} \ . \tag{A.4}$$

We can only accept one root as the correct transformation. Consider the case $y = 0$:

$$x_\pm(0) = -\frac{B}{A} \pm \left|\frac{B}{A} + a\right| \ , \tag{A.5}$$

this should be mapped to $x = a$. For $B/A + a \geq 0$, we therefore need to use the positive root, and for $B/A + a < 0$, we use the negative root.

References

[1] Martino L, Elvira V, Luengo D, Corander J. An Adaptive Population Importance Sampler: Learning from Uncertainty. *IEEE Transactions on Signal Processing*. 2015;63(16): 4422-4437. Available from: doi:10.1109/TSP.2015.2440215 .

[2] Griffiths DJ, Schroeter DF. *Introduction to Quantum Mechanics*. 3rd ed. United Kingdom: Cambridge University Press; 2018.

[3] Weisstein EW. *Erf*. Available from: http://mathworld.wolfram.com/Erf.html [Accessed 16th December 2019].

[4] Uchida Y, Scott M. *Third Year Computational Physics Lecture Notes*. London: Department of Physics, Imperial College London; 2019.

[5] Weinzierl S. *Introduction to Monte Carlo methods*. 2000. Available from: arXiv:hep-ph/0006269 .

[6] Lepage GP. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*. 1978;27(2): 192-203. Available from: doi:10.1016/0021-9991(78)90004-9 .