Preprint SMU-HEP-10-13

Comparison of Algorithms for Monte Carlo Integration of a Multi-dimensional Gaussian Function

Bridget Bertoni
Department of Physics, Southern Methodist University, and
Department of Physics, University of Washington, Seattle
(Dated: August 20, 2010)

Monte Carlo integration of multi-dimensional Gaussian functions is widely applicable in the statistical analysis of functions of many variables, and such analysis is encountered in many fields of science. In this write-up, we compare two Monte Carlo integration algorithms from the Cuba library, Vegas, and Suave, in terms of convergence time and accuracy of evaluation of a multi-dimensional Gaussian integrand. This comparison is important for the analysis of parton distribution functions based on Monte Carlo sampling. In general, we find that Vegas is better suited for this kind of integration. We also conclude that Vegas has faster convergence when used with Mersenne Twister psuedo-random numbers and that Vegas is more accurate when used with Sobol quasi-random numbers.

I. INTRODUCTION

Monte Carlo integration is used as a fast way to numerically estimate integrals which cannot be done analytically. As such, Monte Carlo integration is a widely used technique. It has applications in the fields of statistics, engineering, biology, physics, and finance. Integration of a multi-dimensional Gaussian function has particular importance in statistical data analysis due to the common problem of investigating a function which depends on variables which each have a Gaussian probability distribution. Efficient methods for the integration of such functions find applications in the global QCD analysis and other HEP calculations.

Monte Carlo integration uses random numbers instead of a set grid in order to decide at which points to evaluate the integral. This results in an integral approximation of the form

$$\int_\Omega f(x_1, x_2, ..., x_d) dx_1 dx_2 ... dx_d \approx \frac{1}{N} \sum_{i=1}^{N} f[(x_1, x_2, ..., x_d)_i]. \tag{1}$$

Here the integral is d-dimensional, and i labels the set of d coordinates for a given evaluation point out of N total points. The error in Monte Carlo integration scales like $1/\sqrt{N}$—independent of the dimensionality of the integrand. Hence for many dimensions, Monte Carlo integration converges faster to the exact value of the integral than "classical" integration techniques. (Classical techniques converge more and more slowly with an increase in dimensionality—for instance, the error in using the trapezoidal rule in d dimensions is given by $N^{-2/d} + O((N^{-\frac{1}{d}})^3)$.)

Monte Carlo integration is clearly the better choice when integrating over a large number of dimensions, but even the convergence rate of $1/\sqrt{N}$ becomes slow for the increasingly large number of evaluation points, which is needed when integrating over a large number of dimensions. Because of this, many methods have been developed which hasten the convergence of the integration. In particular, both the introduction of non-random sampling algorithms, which systematically attempt to evaluate the integrand at its most representative points, and alternative types of random numbers used to pick those points, can cause Monte Carlo integration to converge faster.

In the following, two Monte Carlo integration algorithms, Vegas, and Suave, available in the Cuba library of integration algorithms, [2], will be examined in detail for the case of a Gaussian integrand.

Vegas

The Vegas Monte Carlo algorithm was created by G. P. Lepage in [1]. Vegas uses importance sampling to increase convergence. It locates the portions of the integrand that contribute most to the integral, and then it preferentially samples those areas. Vegas initally sets up a uniform grid, consisting of K sections, over the d-dimensional integration area, Ω. From this Vegas creates a uniform probability distribution in which the probability of picking a random point in any given section is $1/K$. Vegas then evaluates the integrand at a given number (N) of random points, chosen according to this uniform probability distribution. Using these evaluated values of the integrand, Vegas creates a new, weighted probability distribution by refining the grid so that the probability for picking a random point in any given section is still $1/K$, but the sections have been resized—weighted by the absolute value of the integrand at the N evaluated points. This ensures that more grid sections are concentrated around places where the integrand is large. Vegas then resamples the integrand using this new probability distribution. With these new integrand samples, Vegas then refines the grid again, and so on, each iteration consisting of a sampling step and a grid refinement.

Using such iterations, Vegas is able to adapt to the integrand, speeding up the convergence of the integral. However it is important to realize that in order to reduce the amount of data to be stored, Vegas uses a weighted probability distribution which is separable into product functions, each of which depend only on one axis variable. Because of this, integration using the Vegas grid is biased towards functions which are aligned with the coordinate axes.

Suave

The Suave Monte Carlo algorithm [2] uses importance sampling and a subdivision strategy to hasten convergence. Suave first samples and evaluates the integrand using a uniform probability distribution. It then locates the region where the integration error is the greatest (based off of a comparison to the user requested relative and absolute accuracies of integration) and bisects this region in the dimension which has the smallest "fluctuation." Roughly speaking, the total fluctuation is calculated from the fluctuation of the individual sample points, F_{ind}. F_{ind} is the absolute value of a product of the relative deviation of the function at the sample points in the region from the region mean, the χ value, and the weight assigned to that point based off of the absolute value of the integrand at the sample points (the same weights are assigned in Vegas). The total fluctation, F_{tot}, is then given by

$$F_{tot} = \left[\sum_i |1 + F_{ind}(x_i)|^p \right]^{2/(3p)}, \tag{2}$$

where the sum is over the sample points in the region, labelled by i, and p is a flatness parameter. In general, a smaller value for the user-defined flatness parameter, p, is better for an integrand with peaks and a larger value is better for a flatter integrand. After the bisection, Suave samples each half of the region separately—sampling more of the half with greater fluctuation (this is why the region is split across the dimension with the *least* fluctuation) and then refines a Vegas grid over the subdivisions. Suave iterates through subdivisions and refinements of the grid until the desired accuracy or maximum number of evaluations is reached. Suave saves the Vegas grid from each iteration, as well as refining it. Because of this Suave uses a lot of memory.

Use of Vegas and Suave in the Cuba Library

The Cuba library [2] is a library of multi-dimensional numerical integration algorithms in Fortran, C, C++, and Mathematica set to integrate over a unit hypercube. It can be found online at http://www.feynarts.de/cuba/. The Cuba library includes both the Vegas and Suave algorithms, additionally allowing the user to choose between Mersenne Twister pseudo-random numbers and Sobol quasi-random numbers for the integration.

Pseudo-random numbers attempt to mimick true random numbers using a (deterministic) computer algorithm. The Mersenne Twister generator is a widely used pseudo-random number generator. Quasi-random (or sub-random) numbers do not attempt to mimick true random numbers, but rather the goal is to produce a uniform distribution. Intuitively one might prefer to use quasi-random numbers for numerical Monte Carlo integration since a uniform sampling (like that used in classical methods) is desirable to thoroughly sample the integrand, yet one still reaps the convergence benefits of using a random sampling (like that used in standard Monte Carlo integration with pseudo-random numbers).

For Monte Carlo integration with quasi-random numbers (known as quasi-Monte Carlo integration) it can be shown that the integration error is proportional to D^*, where D^* is the star discrepancy—a measure of the uniformity of the distribution of the sequence of random numbers within a hypercube. The more uniform the distribution, the lower the discrepancy. For Sobol sequences ([3] and for example see [4]),

$$D^*_{\text{Sobol}} \leq C_d \frac{(\ln\, N)^d}{N} + O\left(\frac{(\ln\, N)^{d-1}}{N}\right). \tag{3}$$

This provides an upper bound on the integration error. It becomes a better estimate for large N and small d. Here

$$C_d = \frac{2^{\tau_d}}{d!(\ln\, 2)^d}\ , \text{ and} \tag{4}$$

$$k\frac{d\,\ln\, d}{\ln\,\ln\, d} \leq \tau_d \leq \frac{d\,\ln\, d}{\ln\, 2} + \frac{d\,\ln\,\ln\, d}{\ln\, 2} + O(d\,\ln\,\ln\, d), \tag{5}$$

where $k > 0$. C_d is always positive and increases faster than exponentially with increasing N.

Also note that in the Cuba library, Vegas and Suave are both deterministic algorithms (fixed by their methods of generating a sequence of random numbers of the form $x_{i+1} = \Phi(x_i)$ and a starting seed, x_0, which is the first number used to begin the sequence of numbers). This means that different runs of Vegas or Suave using the same settings will always produce the same results. This is nice since results are then easy to reproduce exactly, however there is some dependence upon the starting seed which is used.

When using both Vegas and Suave from the Cuba library, the user must specify (among other variables) *ndim*- the number of dimensions of the integral, *ncomp*- the number of components of the integrand, the integrand function, *epsrel* and *epsabs*- the absolute and relative accuracies desired, the type of random numbers, and *mineval* and *maxeval*- the minimum and maximum number of integrand evaluations. Additionally for Vegas, the user must input values for *nstart*- the starting number of integrand evaluations per iteration and for *nincrease*- the increase in the number of integration evaluations per iteration. Suave also requires extra input: *nnew*- the number of new integrand evaluations per subdivision and *flatness*- a value for the flatness parameter.

II. MONTE CARLO INTEGRATION USING VEGAS AND SUAVE FOR A GAUSSIAN TEST FUNCTION

The Cuba library offers a good deal of freedom in parameter choice when using Vegas and Suave, and in practice parameter choice can make a noticeable difference on the convergence of the integration. We have looked at various relationships between the parameters (notably *ndim*, *epsrel*, the type of random numbers, *maxeval*, *nstart*, *nincrease*, *nnew*, and *flatness*) for a Gaussian test function, integrated over a range of standard deviations (σ 's) about its maximum. The integral is given in d dimensions by

$$I_{\text{exact}} = \prod_{i=1}^{d}\left(\frac{1}{\sqrt{2\pi}\sigma_i}\int_{-\bar{N}_i\sigma_i}^{+\bar{N}_i\sigma_i} e^{-\frac{x_i^2}{2\sigma_i^2}}\,dx_i\right), \tag{6}$$

where $\bar{N}_i$ sets the range of integration. In a typical calculation, the exact value of the integral, I_{exact}, was compared with its Monte Carlo approximation, I_{MC}. In all cases, all of the integrand evaluations were used in the final result for the integral (as opposed to the option of using just those from the final iteration).

Vegas

Below are a few graphs which characterize integration of a Gaussian function using the Vegas algorithm from the Cuba library. A summary of the results will be given here.

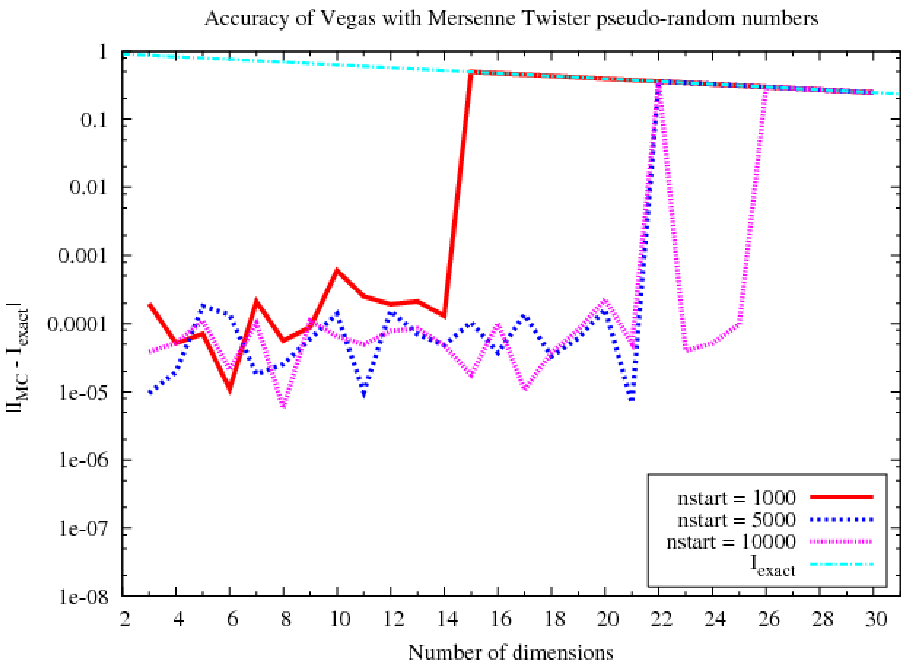

FIG. 1: This plot shows the accuracy obtained after 5 x 10^5 evaluations (*maxeval* = 5 x 10^5 and *epsrel* = *epsabs* = 10^{-22} to ensure this). Whether the end of integration was caused by reaching the maximum number of evaluations or by reaching the targeted accuracy can be confirmed using the variable *fail*. Additional settings were *nincrease* = 0 and $\bar{N} = 2$ for all dimensions.

Fig. 1 shows the accuracy of the Vegas algorithm when using Mersenne Twister random numbers as a function of dimension and the parameter *nstart*. The accuracy is defined in terms of the absolute value of the difference between the exact value of the integral (as determined by Mathematica) and the value of the integral obtained using this Monte Carlo algorithm. The value of the exact integral is graphed for reference—when $|I_{exact} - I_{MC}| = I_{exact}$ it means that the Monte Carlo algorithm reports of value of essentially zero for the integral. This happens since the peak of the Gaussian function becomes sharper and more difficult to sample with many points when integrating over many dimensions. This plot shows that the accuracy for the different values of *nstart* is jumpy, though roughly the same for lower dimensions, and that a higher value of *nstart* allows the peak of the Gaussian function to be “found” for more dimensions.

Fig. 2 is the same plot as Fig. 1 except that Sobol quasi-random numbers have been used instead of Mersenne Twister pseudo-random numbers. In this case, $|I_{exact} - I_{MC}|$ steadily increases for lower dimensions before the settings used are no longer able to sufficiently sample the peak of the Gaussian for higher dimensions. Higher values of *nstart* allow both for the peak to be found in higher dimensions and for consistently better accuracy. Comparing Figs. 1 and 2, one can see that when using Mersenne Twister pseudo-random numbers, Vegas is able to find the peak of the Gaussian in many higher dimensions, though when using Sobol quasi-random numbers, Vegas gives a more accurate value for the integral for the same number of evaluations (in the cases in which it does not integrate to zero).

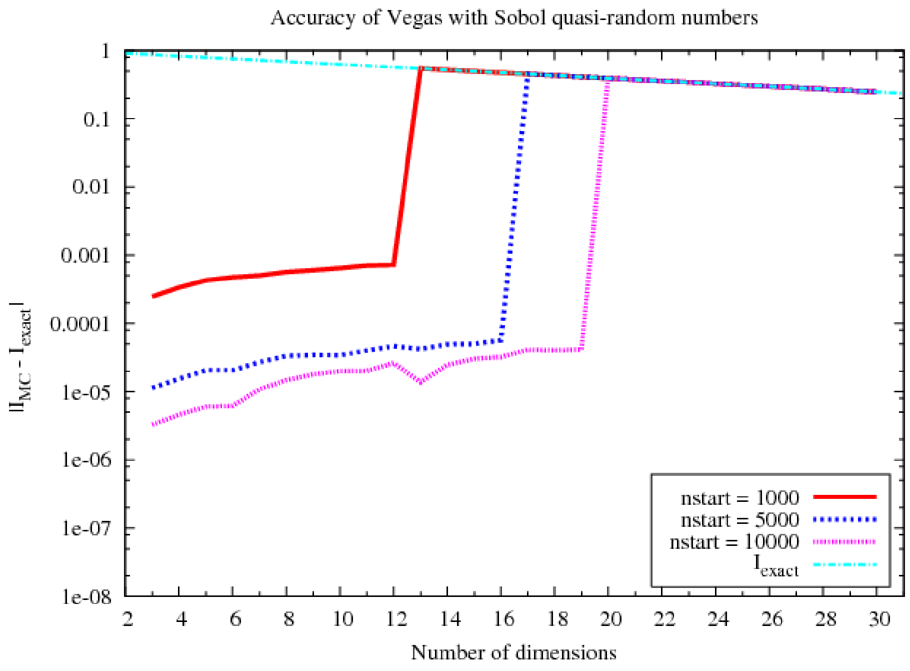

FIG. 2: Again, $maxeval = 5 \text{ x } 10^5$, $epsrel = epsabs = 10^{-22}$, $nincrease = 0$, and $\bar{N} = 2$.

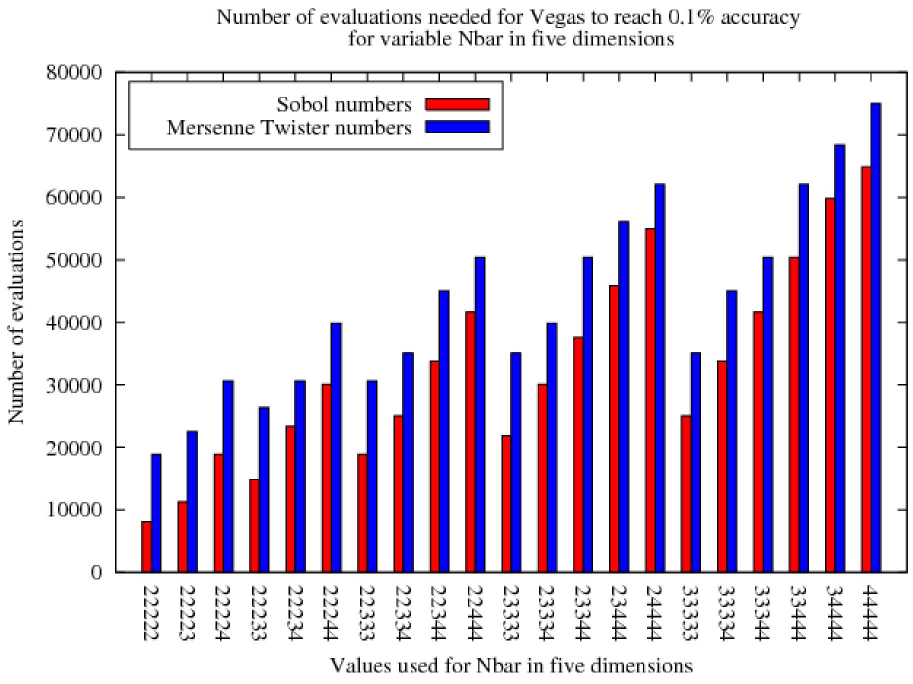

FIG. 3: The horizontal axis contains five number labels which correspond to the five values used for $\bar{N}_i$ in that particular integration. The settings were chosen to be those which required the least number of evaluations for $\bar{N}_i = 3$ for all i for each the Sobol and Mersenne Twister numbers. For Sobol numbers, $nstart = 400$, $nincrease = 50$, and for Mersenne Twister numbers, $nstart = 900$, $nincrease = 300$.

Fig. 3 shows the number of evaluations needed for Vegas to reach 0.1% accuracy in the case of five dimensions, for both Sobol quasi-random numbers and Mersenne Twister pseudo-random numbers, and for several choices of $\bar{N}_i$, where $i = 1, 2, 3, 4, 5$. From the form of the integral in (6), one can see that a higher value of $\bar{N}$ corresponds to a more sharply peaked Gaussian. Since the algorithms in the Cuba library are only set to integrate over the unit hypercube, a variable transformation must be used. The actual integral evaluated in Cuba was

$$\prod_{i=1}^{d} \left(\bar{N}_i \sqrt{\frac{2}{\pi}} \int_0^1 e^{-\frac{\bar{N}_i}{2}(2y_i-1)^2} dy_i \right), \tag{7}$$

which is independent of the σ_i, leaving only $\bar{N}_i$ as the parameter controlling the width of the Gaussian in the i^{th} dimension. It is reasonable to expect more integrand evaluations would be needed to reach the same accuracy for higher values of the $\bar{N}_i$. This is reflected in Fig. 3. Interestingly, at least for the settings used here, integration using the Sobol quasi-random numbers consistently converges to an accuracy of 0.1% using less integrand evaluations than integration with Mersenne Twister pseudo-random numbers.

Fig. 4 shows the settings for *nstart* (the initial number of evaluations) and *nincrease* (the increase in the number of evaluations) in Vegas which minimize the number of evaluations needed to compute the Gaussian integral for variable dimension, $\bar{N}$, and choice of random numbers. It is hard to see many certain trends in the data, but it is clear that when using Mersenne Twister pseudo-random numbers, a larger value for *nstart* is better for faster convergence, regardless of $\bar{N}$.

In Figs. 5 and 6, we show the differing convergence behavior of Sobol quasi-random and Mersenne Twister pseudo-random numbers when using the Vegas algorithm. In both cases the error in the numerical integral is plotted as a function of its predicted dependence on the number of dimensions, d, and the number of evaluations, N.

Fig. 5 shows the case of Sobol quasi-random numbers with the predicted error dependence as given by (3). The six plots are all shown for large N. The change in sign of the slope as d increases occurs because for $d \geq 15$, $d > \ln N$ for the values of N used. The plots are all relatively linear in this large N limit. However it is easy to see that the slopes of the plots in Fig. 5 are all much less than one, while the range for C_d as given by (4) and (5) is certainly greater than one. In particular, from the plots it seems that as $d \to \infty$, $C_d \to 0$, as opposed to $C_d \to \infty$ as predicted by the equations. This disagreement could come from the star discrepancy being an upperbound and not a scaling prediction, or from improved error scaling of the Vegas algorithm due to its use of importance sampling.

Fig. 6 shows the case of Vegas with Mersenne Twister psuedo-random numbers. The predicited error dependence is simply given by $1/\sqrt{N}$ for a standard Monte-Carlo algorithm. These plots similarly show the large N limit but they do not show a linear dependence. This discrepancy could be attributed again to the use of importance sampling in Vegas or could be due to the somewhat erratic behavior of the Mersenne Twister numbers as seen before in Fig. 1 as compared to Fig. 2.

Suave

Below are figures which characterize integration using the Suave algorithm from the Cuba library. Recall that the main difference between Vegas and Suave is the additional calculations of subdivisions that Suave does. Naïvely this should allow Suave to sample integrands more acurately. The next three figures are analogous to the first three figures for Vegas.

Fig. 7 shows the accuracy of the Suave algorithm when using Mersenne Twister pseudo-random numbers. In general, we see that a larger value of *nstart* both gives a more accurate integration and allows for the integration of more dimensions for the same number of evaluations. In Fig. 8, the same is true when using Sobol quasi-random numbers. The use of Sobol random numbers gives roughly a tenfold increase in the accuracy of the integration as opposed to Mersenne Twister random numbers, though for a fixed number of evaluations, integration using Mersenne Twister random numbers is able to give a non-zero value for the Gaussian integral in many more dimensions.

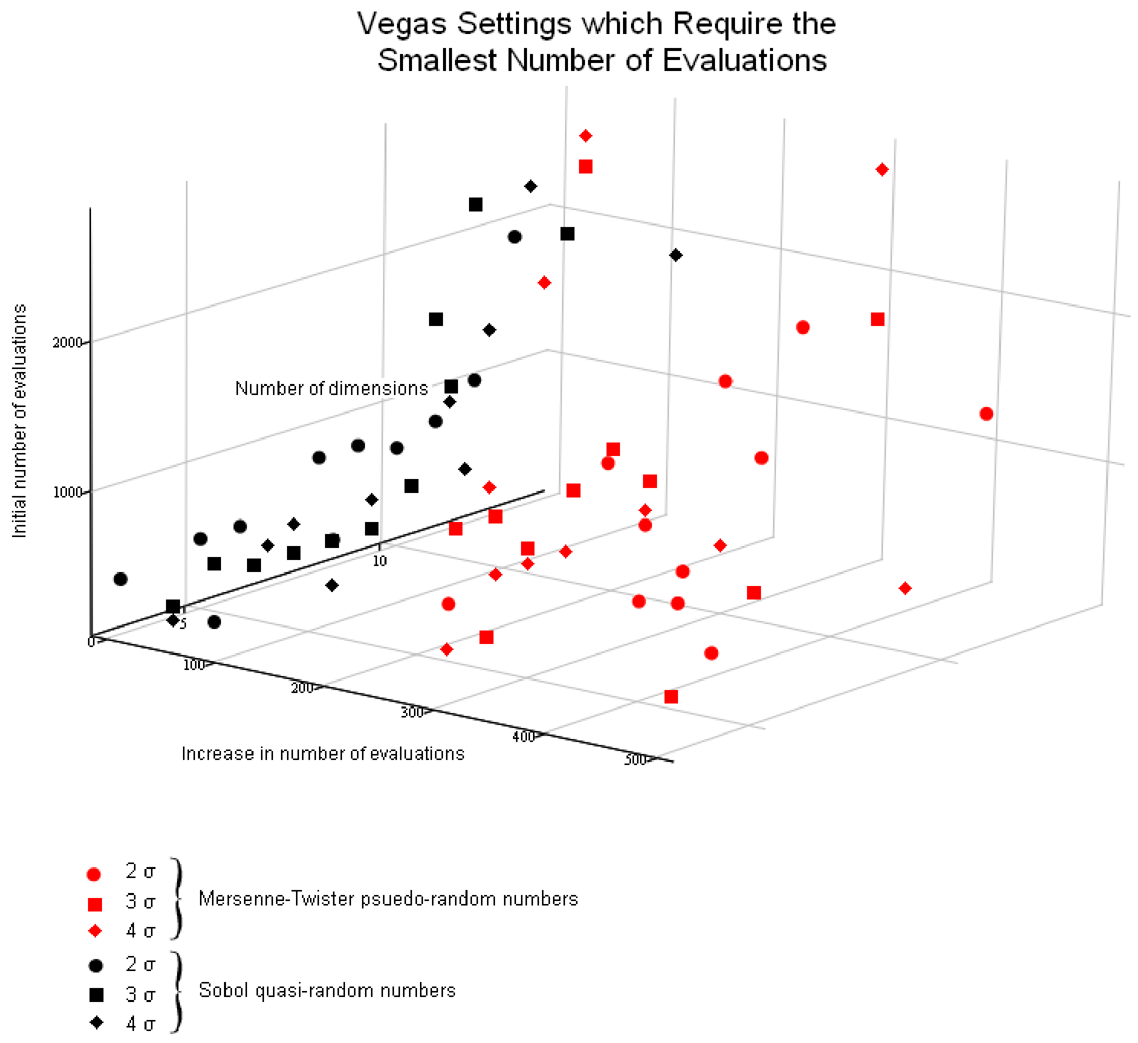

FIG. 4: Here the value for $\bar{N}$ is given in the key using the notation $\bar{N}\sigma$. $\bar{N}$ was the same for all dimensions for any given integration, making the integrand a symmetric Gaussian.

Fig. 9 shows the number of evaluations needed to reach 0.1% accuracy when using Suave in five dimensions. Both Sobol quasi-random numbers and Mersenne Twister pseudo-random numbers were used, and the values for $\bar{N}$ in the five dimensions are specified along the horizontal axis. In all cases, integration using Sobol quasi-random numbers reaches the desired accuracy using a smaller number of evaluations. The data reflects the fact that Suave uses a separable weight function since there is no dependency on a symmetric assortment of the five $\bar{N}_i$—the number of evaluations needed appears to be solely dependent upon the values of $\bar{N}_i$, i.e. it takes more integrand evaluations in order to reach a certain accuracy for larger values of $\bar{N}_i$ (the same goes for Vegas).

Effects of Rotating the Gaussian Integrand

As mentioned in the introduction, both the Vegas and Suave algorithms use a weight function which is separable into a product of functions which each depend only on a single axis variable. For this reason, we noted that this should make the algorithms less efficient when the integrand is not aligned with the integration axes. In order to get an idea of how less efficient these algorithms would become, we used a rotated Gaussian function

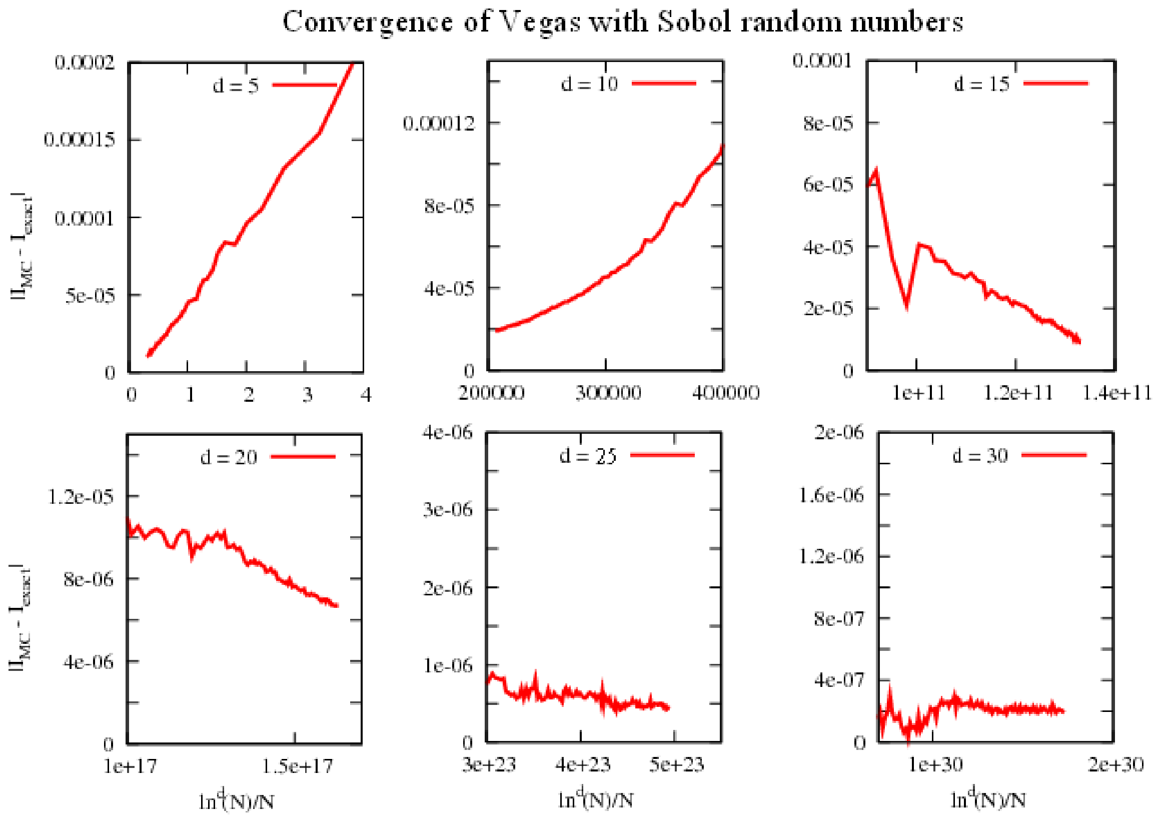

FIG. 5: The following settings were used: for d = 5 and 10, *nincrease* = 50, *nstart* = 1000, *maxeval* $\leq$ 2 x 10^6, for d = 15, *nstart* = 1 x 10^4, for d = 20, *nstart* = 1 x 10^5, *maxeval* $\leq$ 2 x 10^7, for d = 25, *nstart* = 1 x 10^6, *maxeval* $\leq$ 1.5 x 10^8, and for d = 30, *nstart* = 2 x 10^6, *maxeval* $\leq$ 3 x 10^8. The relevant settings which are not given here are the same as the settings specified in the previous dimension.

as the integrand.

An arbitrary rotation in d dimensions was obtained using the Cartan-Dieudonné theorem, which states that a d-dimensional rotation can be decomposed into a product of planar (two-dimensional) rotations. To make this more plausible note that there are $\binom{d}{2} = d(d-1)/2$ different choices for the planar rotation axes and also $d(d-1)/2$ degrees of freedom needed to specify an arbitrary rotation matrix R which fufills $RR^T = I_d$ and $\det(R) = 1$, where I_d is the d-dimensional identity matrix.

For a non-rotated, d-dimensional Gaussian integrand which fits entirely within the unit hypercube, the integral is given by (6). Since the Cuba library is constrained to integration over the unit hypercube, for integration of a Gaussian integrand after it has been rotated, the longest diagonal of the rotated hypercube must fit within a non-rotated hypercube (defined by the integration axes). In order to enforce this, each side of the rotated hypercube (previously equal to one) must be scaled by a factor of $1/\sqrt{d}$, where d is the number of dimensions of integration. The final form for the integral is given by

$$\prod_{i=1}^{d} \left\{ \sqrt{\frac{2d}{\pi}} \bar{N}_i \int_{\frac{1}{2}-\frac{1}{2\sqrt{d}}}^{\frac{1}{2}+\frac{1}{2\sqrt{d}}} \exp\left[-2N\bar{N}_i^2 \left(z_i - \frac{1}{2} \right)^2 \right] dz_i \right\}. \tag{8}$$

Here the z_i are defined by the axes of the hypercube which has been rotated with respect to the axes of integration. The d-dimensional rotation is carried out simply by taking the product of rotations in every possible set of two-dimensional axes.

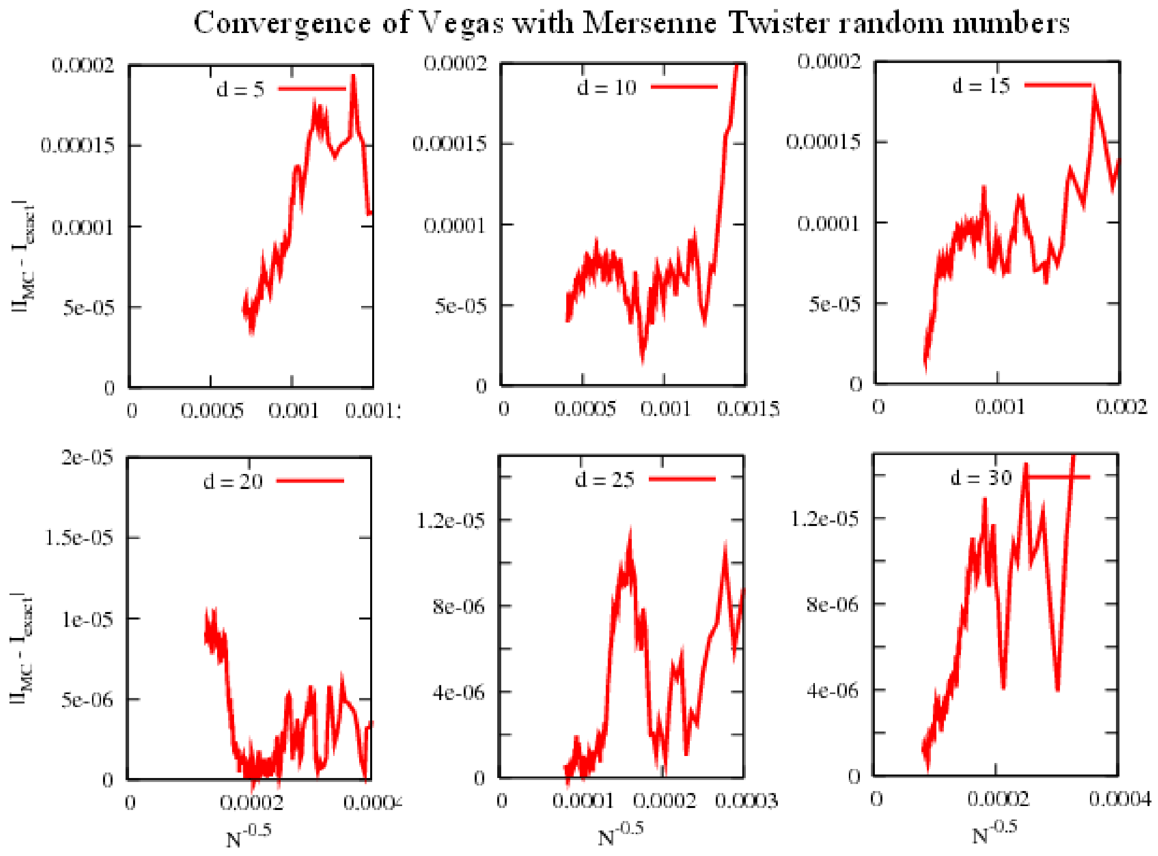

FIG. 6: The following settings were used: for d = 5, *nincrease* = 50, *nstart* = 1000, *maxeval* $\leq$ 2 x 10^6, for d = 10, *maxeval* $\leq$ 6 x 10^6, for d = 15, *nstart* = 1 x 10^4, for d = 20, *nstart* = 1 x 10^5, *maxeval* $\leq$ 6 x 10^7, and for d = 25 and 30, *nstart* = 1 x 10^6, *maxeval* $\leq$ 1.5 x 10^8. The relevant settings which are not given here are the same as those specified in the previous dimension.

As can be seen from (8), as the number of dimensions, d, increases, the area over which the integrand is nonzero decreases as $d^{-d/2}$. This makes it even harder for the Cuba integration algorithms to converge for a large number of dimensions, but such difficulties are unavoidable if one wishes to compare arbitrary d-dimensional rotations. Thus, in order to examine the effects of rotation on the integration of a Gaussian, it is easiest to look at effects in a smaller number of dimensions. Six- and eight-dimensional rotation examples are pictured in Fig. 10.

Fig. 10 clearly shows that a rotation of an asymmetric Gaussian integrand causes the number of evaluations needed to reach 0.1% accuracy to increase. Additionally, the two graphs support the conclusion that the Vegas algorithm is better suited for integration of a rotated Gaussian function and that the number of evaluations needed is comparable when using Vegas with either Sobol or Mersenne Twister random numbers. Of course these results depend on the values of the various input parameters, but in general, additional results using higher dimensions and different values for the $\bar{N}_i$ support the results shown in Fig. 10 and the conclusions drawn above.

Integration over a Large Number of Dimensions

In this section we consider integration using Vegas and Suave with either Sobol or Mresenne Twister random numbers for a large number of dimensions. Mulitple settings for dimensions d = 20, 25, and 30 with either $\bar{N}_i = 2$ for all dimensions or $1 \leq \bar{N}_i \leq 3$ were examined.

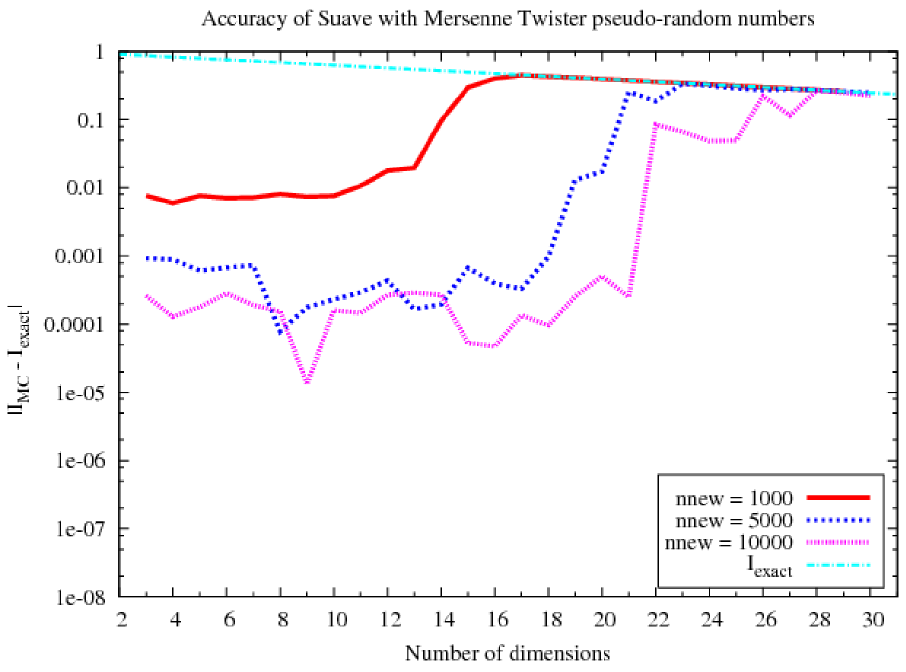

FIG. 7: This plot shows the accuracy obtained after 5 x 10^5 evaluations (*epsrel* $=$ *epsabs* $= 10^{-22}$ to ensure this). Additional settings were *flatness* $= 1$ and $\bar{N} = 2$ for all dimensions.

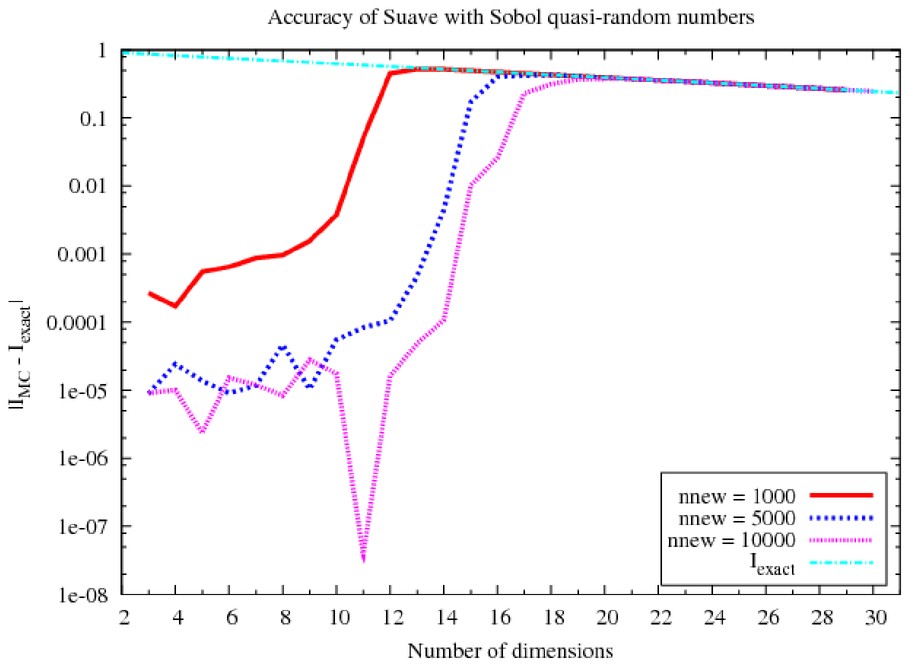

FIG. 8: The settings are the same as those in Fig. 7: *maxeval* $= 5$ x 10^5, *epsrel* $=$ *epsabs* $= 10^{-22}$, *flatness* $= 1$, and $\bar{N} = 2$.

The data indicates that Vegas with Mersenne Twister random numbers is most likely to converge. When Suave with Mersenne Twister random numbers is used—and the integration converges—it converges in a number of

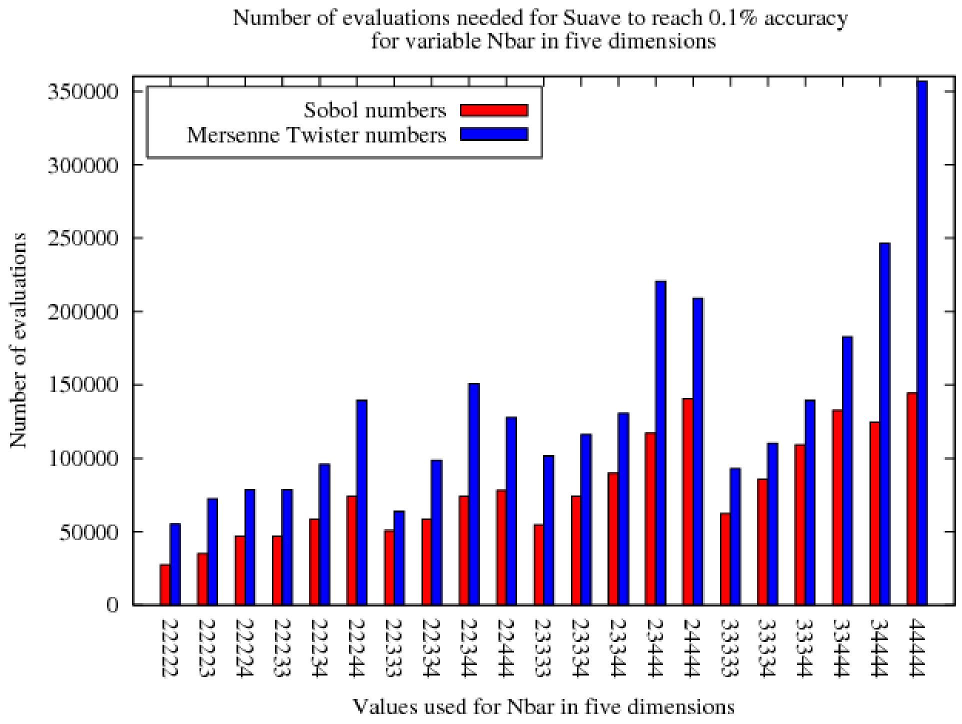

FIG. 9: The best settings for five dimensions, $\bar{N} = 3$ for all dimensions, and *flatness* ≥ 1 were used. For Sobol numbers, *nnew* $= 3900$, *flatness* $= 1$, and for Mersenne Twister numbers, *nnew* $= 2900$, *flatness* $= 1$.

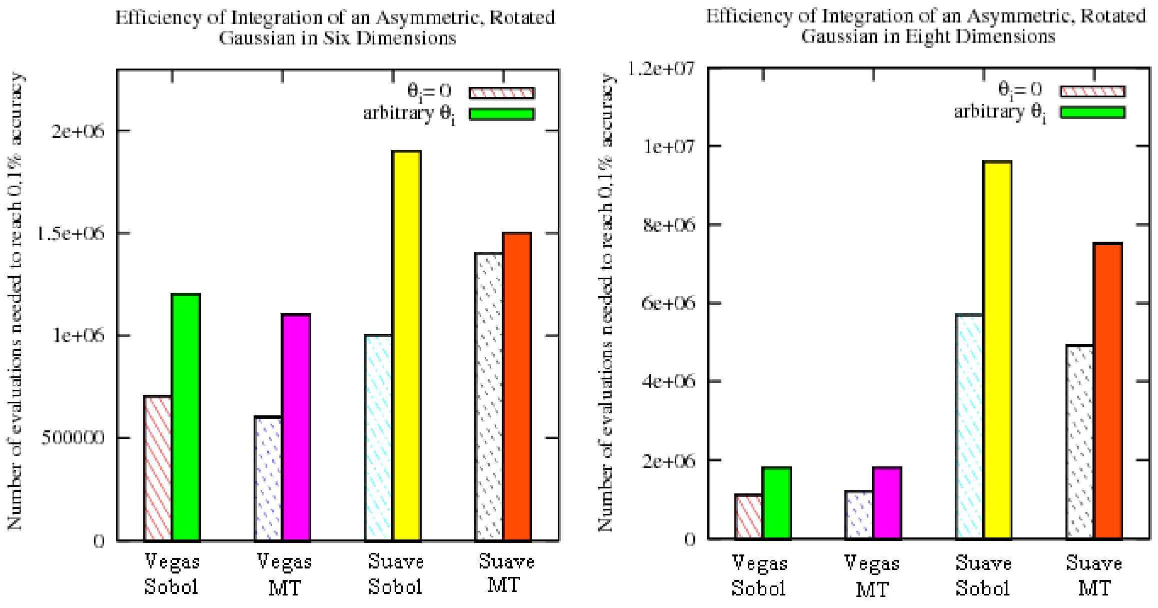

FIG. 10: These figures show the results of integration using four different integration methods in the Cuba library with a rotated two-dimensional Gaussian integrand. The Gaussian integrand was asymmetric, with $\bar{N}_1 = 3,\ \bar{N}_2 = 1,\ \bar{N}_i = 2$, where $3 \leq i \leq 6$ or 8. Relevant settings used were *maxeval* $= 5 \text{ x } 10^8$, *nnew* $=$ *nstart* $= 1 \text{ x } 10^5$, *epsrel* $= 1 \text{ x } 10^{-3}$, and *epsabs* $= 1 \text{ x } 10^{-22}$. Rotation angles were generated randomly. MT is shorthand for Mersenne Twister random numbers.

evaluations comparable to that used by Vegas with Mersenne Twister numbers. Both the Vegas and Suave integration algorithms with Sobol random numbers converge less often and take either the same amount of evaluations or more to converge.

Additional problems occur when a large number of dimensions is used in the integration. For instance, the algorithm may incorrectly decide that the integral has converged when in reality it has not. This happens for both Vegas and Suave, though it is more common in Suave. The user should also be aware that Suave will frequently run out of memory when doing integration over a large number of dimensions, though once this is recognized, Suave can just be given a larger amount of memory to run on from the beginning, and one does not have to waste time waiting for integrations to converge which will ultimately end prematurely due to a memory shortage.

III. CONCLUSION

Comparing the descriptions of Vegas and Suave in the sections above, some general statements can be made about the Vegas and Suave algorithms when integrating a Gaussian function. In the case of an arbitrarily rotated (or aligned), Gaussian integrand, integration using the Suave algorithm is NOT better for “finding” the peak of the multi-dimensional Gaussian—this can be seen from the accuracy plots (Figs. 7 and 8) which gradually approach the point at which the algorithm integrates to zero. The analogous figures for the Vegas algorithm (Figs. 1 and 2) abrubtly switch from integration with good accuracy to integration which does not sufficiently sample the peak of the Gaussian and integrates to zero. The dimensions at which the integration results in zero for both Vegas and Suave (for the same type of random numbers) are roughly the same.

A comparison of the vertical axes in Figs. 3 and 4 shows that in the case of a Gaussian integrand which is aligned with the integration axes, to integrate to a desired accuracy, many more evaluations ($\sim$ CPU time) are needed when using the Suave algorithm, as opposed to Vegas. Suave also generally needs more integrand evaluations to integrate a rotated Gaussian integrand (e.g. see Fig. 10). This is somewhat unexpected, since the Suave algorithm specifically pinpoints the portions of the integrand which fluctuate the most, so one would think that integration using the Suave algorithm would quickly pick up on the peak of the Gaussian test function and produce an accurate result of the integral faster than the Vegas algorithm. But as in the previous paragraph, this is not quite the case.

It should be noted that is possible that there are better settings for Suave (namely for $0 < \textit{flatness} < 1$) which would make it perform better. It is also possible that the subdivisions implemented in Suave are more beneficial when the integrand fluctuates a lot (unlike the multi-dimensional Gaussian which only has a single peak). Additionally, Suave saves all of its grids, so it uses a lot of memory when it runs—often causing the integration to end prematurely when it runs out of memory. This is a particular hindrance for integration over many dimensions, making it worse candidate for high-dimensional integration of a Gaussian function.

The information here indicates that in the case of an arbitrarily rotated and multi-dimensional Gaussian integrand, Vegas is definitely a more efficient algorithm for integration. For a given choice of input parameters, when Vegas is used with Mersenne Twister random numbers, the integration is more likely to converge and converge faster (reaching a given integration accuracy using a smaller number of evaluations). When Vegas is used with Sobol random numbers, the integration is less likely to converge, but when it does coverge it generally gives a more accurate value for the integral. Yet even knowing this, in practice it is still important to use various numerical integration methods to add confidence to the answer that any one method returned for the integral.

Acknowledgements

This work was done as a part of a 2010 summer research project in the Physics Department at Southern Methodist University in Dallas, TX. It was supported by P. Nadolsky’s start-up grant. The author thanks P. Nadolsky for his guidance in this work.

[1] G. P. Lepage, J. Comp. Phys. 27 (1978) 192.

[2] T. Hahn, Comp. Phys. Comm. 168 (2005) 78-95.

[3] Sobol', I. M., U.S.S.R. Comp. Math. and Math. Physics 7, 4 (1967) 86-112.

[4] S. Weinzierl, arXiv:hep-ph/0006269v1.