A Comparison between (Quasi-)Monte Carlo and Cubature Rule Based Methods for Solving High-dimensional Integration Problems

Rudolf Schürer

Department of Mathematics, University of Salzburg, AUSTRIA

Abstract

Algorithms for estimating the integral over hyper-rectangular regions are discussed. Solving this problem in high dimensions is usually considered a domain of Monte Carlo and quasi-Monte Carlo methods, because their power degrades little with increasing dimension. These algorithms are compared to integration routines based on interpolatory cubature rules, which are usually only used in low dimensions. Adaptive as well as non-adaptive algorithms based on a variety of rules result in a wide range of different integration routines. Empirical tests performed with Genz’s test function package show that cubature rule based algorithms can provide more accurate results than quasi-Monte Carlo routines for dimensions up to $s = 100$.

Key words: Numerical integration, Monte Carlo, Quasi-Monte Carlo, Cubature rule

1991 MSC: 65C05, 65D30, 65D32

1 Introduction

We investigate the problem of estimating an approximation

$$Q_n f = \sum_{i=1}^{n} w_i f(\bar{x}_i) \quad \text{for the integral} \quad I f := \int_{C_s} f(\bar{x}) \, d\bar{x},$$

with $f: C_s \to \mathbb{R}$ and $C_s$ denoting an $s$-dimensional, hyper-rectangular region. Numerical integration in high dimensions is usually considered to be a typical application of Monte Carlo and quasi-Monte Carlo routines, with accurate

Email address: rudolf.schuerer@sbg.ac.at (Rudolf Schürer).
URL: http://www.cosy.sbg.ac.at/~rschuer/ (Rudolf Schürer).

Preprint submitted to Elsevier Science 25th March 2003
and reliable results being reported for dimensions up to $s = 300$ and beyond [1]. On the other hand, it is well known that for integration in low dimensions, routines based on interpolatory cubature rules turn out to deliver more accurate estimations if the integrand is sufficiently smooth.

However, little work has been done to compare both types of algorithms in dimensions above, say $s = 10$.\footnote{One exception is [2], which gives empirical results up to $s = 40$.} This article tries to close this gap for dimensions up to $s = 100$.

## 2 Construction of Integration Routines

Three widely used, but distinctly different algorithms for numerical integration are evaluated: Monte Carlo methods, quasi-Monte Carlo methods, and methods based on interpolatory cubature rules. With interpolatory rules, adaptive as well as non-adaptive algorithms can be constructed.

This section discusses all four types of algorithms, discusses their implementation and gives a short overview on their theoretical advantages and disadvantages.

### 2.1 Monte Carlo Routines

Numerical integration based on Monte Carlo routines uses an integration formula of the form

$$Q_n f = \frac{1}{n} \sum_{i=1}^{n} f(\bar{x}),$$

with $\bar{x}_1, \ldots, \bar{x}_n \in \mathbb{R}^s$ denoting $n$ independent samples of a random variable with uniform distribution in $C_s$.

Based thereupon, the \textit{Strong Law of Large Numbers} guarantees an almost certain convergence of the approximation for $n \to \infty$. In addition to that, the \textit{Central Limit Theorem} shows that a rate of convergence of $O(\frac{\sigma(f)}{\sqrt{n}})$ is obtained, with $\sigma(f)$ denoting the standard deviation of $f$. Therefore, the convergence rate of Monte Carlo integration is independent of the dimension $s$, and the same accuracy can be expected in all dimensions for integrands with the same standard deviation.

To simulate the required random process on a computer, a pseudo random number generated (PRNG) has to be used. For our implementation, we use the \textit{Mersenne Twister} [3], which is widely believed to be one of the best
PRNGs known today. It has a period length of $2^{19937} - 1$ (a Mersenne Prime), produces a sequence that is 623-dimensionally equidistributed up to 32 bits, and has passed many stringent tests, including the die-hard and the load test.

2.2 Quasi-Monte Carlo Routines

Quasi-Monte Carlo routines are similar to Monte Carlo algorithms. However, instead of using samples from a random variable, a low-discrepancy sequence is used for generating the abscissa set. For this kind of algorithms, the Koksma-Hlawka-Inequality guarantees a rate of convergence of $O\left(\frac{\log^s n}{n}\right)$ for all integrands with bounded variation in the sense of Hardy and Krause. Even though the term $\log^s n$ dominates the convergence order for reasonable values $s$ and $n$, empirical results show that a convergence rate of $O\left(\frac{1}{n}\right)$ is obtained for many integrand types. Therefore, quasi-Monte Carlo routines are well suited for high dimensions.

In this work, digital sequences due to Sobol [4] and Niederreiter [5] (in base $b = 2$) are used. We add an additional equidistributed coordinate, which allows us to use an $s - 1$ dimensional sequence to generate an $s$-dimensional abscissa point set. For both sequences, the initial point (the corner $\bar{x}_0 = (0, \ldots, 0)$ of the unit cube) is skipped, which increases the quality of the generated (finite) point-set in both cases. Doing so adds an additional desirable feature to the point-set generated by Sobol’s sequence: The points $\bar{x}_1, \ldots, \bar{x}_{2^k - 1}$ projected onto any coordinate axis are always symmetric with center $1/2$, which, for instance, makes the resulting formula exact for all linear integrands.

2.3 Methods based on Interpolatory Cubature Rules

Interpolatory cubature rules [6] are integration formulas of the form

$$Q_m f = \sum_{i=1}^{m} w_i f(\bar{x}_i)$$

such that all multi-variate polynomials up to a certain degree $d$ are integrated exactly, i.e. $I p = Q_m p$ for all polynomials $p \in \mathbb{P}_d^s$. The number $m$ of abscissas can not be chosen freely, but depends on the structure of the formula. Usually, $m$ increases rapidly with the dimension $s$ and the desired degree $d$.

Due to this definition, integration routines based on cubature rules can be expected to be most efficient for integrands that can be approximated by (low degree) polynomials, i.e. for smooth integrands.
Table 1
Implemented cubature rules

| Name              | Degree | $m$   | $\sum_{i=1}^{m} |w_i|$ | Reference |
|-------------------|--------|-------|----------------|-----------|
| Octahedron        | 3      | $O(s)$| 1              | [7]       |
| Hammer & Stroud   | 5      | $O(s^2)$| 0.62$s^2 + O(s)$| [8]       |
| Stroud            | 5      | $O(s^2)$| 1.4$s^2 + O(s)$| [9]       |
| Phillips          | 7      | $O(s^3)$| 0.23$s^3 + O(s^2)$| [10]      |
| Stenger           | 9      | $O(s^4)$| 1.2$s^4 + O(s^3)$| [11,6]    |
| Genz & Malik      | 7      | $O(2^s)$| 0.041$s^2 + O(s)$| [12]      |

The construction of high-dimensional interpolatory cubature rules is a difficult task. Even though a large number of theoretical results are available today, building rules with a low number of abscissas remains an art.

Table 1 lists the implemented cubature rules. It covers most known rules that can be constructed for arbitrary dimensions $s$. In addition to its name and the degree $d$, the table lists the order of magnitude of the number of abscissas, as well as the sum of the absolute values of the weights $w_i$. The latter value is 1 for rules with positive weights only, but increases for rules that have negative weights. It can be used as a figure of merit to rate the quality of the rule.

For a given dimension $s$, cubature rules have a fixed number $m$ of abscissas. Therefore, integration routines based on a rule $Q_m$ have to split $C_s$ into $k := \left\lfloor \frac{n}{m} \right\rfloor$ rectangular subregions and apply $Q_m$ to each of them, resulting in a total number of $km \approx n$ abscissas. As we have outlined at the beginning of Section 2, there are two distinctly different approaches for performing this region split: Adaptive (Section 2.3.2) as well as non-adaptive routines (Section 2.3.1).

2.3.1 Non-Adaptive Routines

Non-adaptive integration routines split $C_s$ into $k$ (approximately) equally sized subcubes in an a-priori determined, regular way. This can be done, for instance, by splitting $C_s$ either $\left\lfloor \sqrt[k]{k} \right\rfloor$ or $\left\lceil \sqrt[k]{k} \right\rceil$ times along each coordinate axis.

The integral approximation $Qf$ is obtained by applying $Q_m$ to each resulting subcube and by summing up the obtained results.

2.3.2 Adaptive Algorithms

Instead of using a predetermined subregion structure, adaptive algorithms calculate a subdivision tree that is tailor-made for the given integrand. They
Figure 1. Abscissa set created by an adaptive algorithm for a discontinuous integrand

proceed by applying the cubature rule successively to smaller subregions of the original integration domain. The selection of these subregions adapts to “difficult” areas in the integration domain by refining subregions with large estimated errors. Algorithm 1 describes this procedure in more detail.

Algorithm 1 Adaptive integration

Put $C_s$ into region collection

while estimated error too large do

Choose subregion with largest error

Split this region

Apply basic rule to newly created regions

Store new regions in region collection

end while

Figure 1 compares the abscissa set created by an adaptive algorithm applied to a discontinuous integrand (right side) to a plot of the corresponding function. It can be seen that the algorithm focuses its integrand evaluation on problematic regions in the integration domain.

An algorithm as outlined above requires some method for determining the subregion with the most unreliable result. This can be done by estimating the integration error for each subregion in addition to approximating the integral. The most common approach for getting this error estimation is to use two interpolatory cubature rules

$$Q^{(1)} f = \sum_{i=1}^{m^{(1)}} w_i^{(1)} f(x_i^{(1)}) \quad \text{and} \quad Q^{(2)} f = \sum_{i=1}^{m^{(2)}} w_i^{(2)} f(x_i^{(2)})$$

with $\deg Q^{(1)} > \deg Q^{(2)}$. Under these conditions, $Q^{(1)}$ can be used as an approximation of the integral $\int f$, while the integration error can be estimated by $E f := \left| Q^{(1)} f - Q^{(2)} f \right|$. Due to the fact that the degree of $Q^{(2)}$ is smaller than

\[
Q^{(1)} f = \sum_{i=1}^{m^{(1)}} w_i^{(1)} f(x_i^{(1)}) \quad \text{and} \quad Q^{(2)} f = \sum_{i=1}^{m^{(2)}} w_i^{(2)} f(x_i^{(2)})
\]
Table 2
Cubature rule pairs

<table>
<thead>
<tr>
<th>Name</th>
<th>$Q^{(1)}$</th>
<th>$Q^{(2)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-3</td>
<td>Hammer &amp; Stroud</td>
<td>Octahedron</td>
</tr>
<tr>
<td>7-5-5</td>
<td>Phillips</td>
<td>Hammer &amp; Stroud + Stroud</td>
</tr>
<tr>
<td>9-7</td>
<td>Stenger</td>
<td>Phillips</td>
</tr>
<tr>
<td>GM</td>
<td>Genz &amp; Malik</td>
<td></td>
</tr>
</tbody>
</table>

that of $Q^{(1)}$, and therefore $m^{(2)}$ is significantly smaller than $m^{(1)}$, estimating the integration error comes at little additional cost. In some cases, the abscissas of $Q^{(2)}$ are even a subset of the abscissas of $Q^{(1)}$. For these *embedded rules* no extra integrand evaluations are required to estimate the integration error.

The cubature rule pairs $Q^{(1)}$-$Q^{(2)}$ used here are listed in Table 2. Formula 7-5-5 is a special case, because it uses 2 additional basic rules $Q^{(2)}$ and $Q^{(2)}$ for error estimation: $Ef$ is calculated by the formula

$$Ef := \max \left\{ |Q^{(1)} f - Q^{(2)} f|, |Q^{(1)} f - \overline{Q}^{(2)} f| \right\}.$$

Genz & Malik does already contain an embedded fifth degree rule for error estimation, so no additional basic rule is required for GM.

3 Testing

Numerical tests are performed for all discussed algorithms for dimensions ranging from $s = 2$ up to 100. The number of allowed integrand evaluations is raised by a factor of 2 up to a maximum of $2^{25} \approx 33$ millions. We use the test function package proposed by Genz [13], which defines six function families, each of them characterized by some peculiarity (see Table 3). For each family, 20 instances are created by choosing random unaffective and affective parameters $u_i$ and $a_i$. The vector $\vec{a} = (a_1, \ldots, a_s)$ of affective parameters is scaled such that $||\vec{a}||_1$ meets the requested difficulty specified in Table 3.

For each instance $k = 1, \ldots, 20$ of an integrand family, the number of correct digits $d_k$ (relative to the average magnitude of the integral of the current function family) is calculated by the formula

$$d_k = -\log_{10} \frac{|I_k - Q_k f_k|}{\sum_{i=1}^{20} |I_i|} \quad \text{for } k = 1, \ldots, 20.$$

\footnote{Two test integrands can not be computed in dimensions exceeding 20 and 40, respectively. So these results are missing in the final discussion.}
Table 3
Genz’s test integrand families

<table>
<thead>
<tr>
<th>Integrand Family</th>
<th></th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1(\vec{x}) := \cos(2\pi u_1 + \sum_{i=1}^{s} a_i x_i)$</td>
<td>$|a|_1$</td>
<td>Oscillatory</td>
</tr>
<tr>
<td>$f_2(\vec{x}) := \prod_{i=1}^{s} \frac{1}{\sqrt{a_i^2 + (x_i - u_i)^2}}$</td>
<td>$\frac{600}{s^2}$</td>
<td>Product Peak</td>
</tr>
<tr>
<td>$f_3(\vec{x}) := (1 + \sum_{i=1}^{s} a_i x_i)^{-(s+1)}$</td>
<td>$\frac{600}{s^2}$</td>
<td>Corner Peak</td>
</tr>
<tr>
<td>$f_4(\vec{x}) := \exp\left(- \sum_{i=1}^{s} a_i^2 (x_i - u_i)^2\right)$</td>
<td>$\frac{100}{s}$</td>
<td>Gaussian</td>
</tr>
<tr>
<td>$f_5(\vec{x}) := \exp\left(- \sum_{i=1}^{s} a_i</td>
<td>x_i - u_i</td>
<td>\right)$</td>
</tr>
<tr>
<td>$f_6(\vec{x}) := \begin{cases} 0 &amp; x_1 &gt; u_1 \lor x_2 &gt; u_2 \ \exp(\sum_{i=1}^{s} a_i x_i) &amp; \text{otherwise} \end{cases}$</td>
<td>$\frac{100}{s^2}$</td>
<td>Discontinuous</td>
</tr>
</tbody>
</table>

Based on these values derived for each test integrand instance, statistical methods are used to evaluate the integrand family. The following charts show mean values with error bars based on standard deviation.

4 Results

Which algorithm performs best depends highly on the integrand and its dimension. While (quasi-)Monte Carlo degrades little for high dimensions and non-smooth integrands, the domain of cubature rule based algorithms is clearly that of smooth, low-dimensional integrands. In addition to that, properties of the cubature rule have to be taken into account for rule-based algorithms, because each rule has its specific strengths and weaknesses for certain dimensions and integrand types.

Figure 2 shows the best algorithm for each examined dimension and integrand type. The quasi-Monte Carlo algorithm is denoted by a slanted pattern, while cubature rule base algorithms are depicted by shaded areas, with an additional vertical pattern if the performance of the non-adaptive algorithm exceed that of the adaptive routine. Monte Carlo integration does not show up because it is always inferior compared to one of the other algorithms.

If two algorithms are reported as “best” for a problem, the one in the major part of the field achieves the best performance, while the one in the lower right corner can be expected to beat the first one eventually if the number of integrand evaluations is increased beyond $2^{25}$.
4.1 Detailed Discussion

First, we evaluate the results for the $C^0$ integrand, where the typical behavior for a non-smooth function can be seen: Cubature rule based algorithms are completely outperformed by quasi-Monte Carlo in all but the lowest dimensions. The left chart in Figure 3 shows the break-even point $s = 5$; the adaptive algorithm based on 7-5-5 can almost catch up with Sobol. With increasing dimensions, however, quasi-Monte Carlo is the only reasonable approach.

For the discontinuous integrand, we would expect an even bigger advantage for the quasi-Monte Carlo algorithm. However, as it is noted in Figure 2, adaptive algorithms are superior up to $s = 40$! The right chart in Figure 3 shows the details for $s = 30$. While quasi-Monte Carlo and Monte Carlo show a steady increase in accuracy, the adaptive algorithm improves its accuracy hyper-linearly and overtakes. This behavior can easily be explained by a careful evaluation of both test integrals: While Genz Function 6 is discontinuous only on two hyperplanes, the $C^0$ function is non-differentiable on $s$ hyperplanes, making it significantly harder for adaptive algorithms to find and to adapt
to the difficult regions. It should also be noted that for both integrands, the non-adaptive routines show no significant accuracy improvement at all.

The results for Oscillatory and Product Peak test integrands are rather different compared to the first two cases. Both functions are smooth, and cubature rule based algorithms outperform Monte Carlo and quasi-Monte Carlo for most dimensions up to $s = 100$. However, it should be noted that the optimal cubature rule changes: While high-degree rules work well in low dimensions, they have to be replaced by lower degree, but cheaper rules when the dimension increases. Non-adaptive routines seem to be most successful at the upper end of the range of a certain cubature rule. Figure 4 shows examples for the Oscillatory integrand with $s = 100$ and the Product Peak function with $s = 40$. It can be seen that quasi-Monte Carlo routines have a higher convergence rate in these high dimensions. Cubature rule based routines, however, start off with a much better approximation requiring significantly less points.

5 Conclusion

Choosing an integration routine solely based on the dimensionality of the problem is impossible. Worst case error bounds in high dimensions show that there is no single algorithm that can cope with arbitrary high-dimensional integrands. Therefore, the only hope for solving these problems successfully is to choose an integration routine based on known properties of the integrand. And this routine might turn out to be a cubature rule based algorithm as well as a quasi-Monte Carlo routine.

6 Acknowledgments

This work is supported by the Austrian Science Foundation Grant no. S8311-MAT and by Österreichische Nationalbank, Jubiläumsfonds project no.\_6788.
References


