



Quasi-Monte Carlo quadratures for multivariate smooth functions

Sylvain Maire*, Christophe De Luigi

ISITV, Université du Sud Toulon-Var, Avenue G. Pompidou BP 56, 83262 La Valette du Var cedex, France

Available online 8 April 2005

Abstract

We compute approximations of multivariate smooth functions by fitting random and quasi-random data to reduced size Tchebychef polynomial approximation models. We discuss the optimization of the data used in the least square method by testing several quasi-random sequences. Points built from optimal quadratic quantization are especially efficient. Very accurate approximation type quadrature formulas are obtained avoiding the usual periodisation problems when using lattices rules. Some numerical tests confirm the efficiency of our quadratures compared to standard methods.

© 2005 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Quadrature formulas; Polynomial approximations; Least square method; Quasi-random sequences; Quantization

1. Introduction

The problem of the numerical integration of Q -dimensional smooth functions over an hypercube $D = [0, 1]^Q$ has been widely studied. In very high dimensions, crude Monte Carlo integration is often considered as the only possibility as its accuracy $\frac{\sigma}{\sqrt{N}}$ using N random points depends only on the variance $\sigma_f^2 = \int_D f^2(x) dx - (\int_D f(x) dx)^2$ of the integrand [6,12–14]. In low dimensions, say less than 3 or 4, Gauss product rules appear as a reasonable choice. Between these two bounds, a lot of tools have been developed to take into account the dimensional effect. In rather high dimensions, quasi-Monte Carlo sequences [13,19,24,27] like Halton, Faure or Sobol sequences are very interesting as their speed of con-

* Corresponding author.

E-mail addresses: maire@univ-tln.fr (S. Maire), deluigi@univ-tln.fr (C. De Luigi).

vergence is a $O(\frac{\ln(N)^{Q-1}}{N})$. Our work here concerns moderate dimensions, say less than 10. In this range, quadrature formulas based on approximations on different kinds of reduced size bases have shown their efficiency. Concerning the Q -dimensional Fourier bases on periodic functions, Korobov spaces [11,13] have been introduced. They rely on a decay of the Fourier coefficients a_m as

$$|a_m| \leq \frac{C}{(\tilde{m}_1 \tilde{m}_2 \cdots \tilde{m}_Q)^\alpha},$$

where $\tilde{m} = \max(1, |m|)$ and $\alpha > 1$ is linked to the regularity of the integrand. To achieve such a decay for non-periodic functions, one has to transform the original integral using the periodisation method [2,10] so that the periodised function verifies the same decay. This enables one to build integration formulas known as lattice rules [11,13,25] which try to annihilate the most significant Fourier coefficients belonging to Zaremba crosses $Z_{Q,d}$ defined as

$$Z_{Q,d} = \{m \in \mathbb{Z}^Q \mid (\tilde{m}_1 \cdots \tilde{m}_Q) = d\}.$$

Sloan and Kachoyan [25] gave an estimation of the accuracy of these lattice rules which is mainly described by $O(\frac{1}{\rho(L)^\alpha})$ where

$$\rho(L) = \min\{d: L^\perp \cap Z_{Q,d} \neq \emptyset\}$$

for a given integration lattice L . The main drawback of the periodisation method is an artificial effect on the constant C which grows very quickly with α . This is especially true in high dimension [6] so that α is usually taken equal to 2 or 3. In the case of polynomial approximations, Novak and Ritter [20] have introduced an approach based on the combination of the Clenshaw–Curtis rule and on Smoliak's construction [26]. They obtain exact quadrature formulas on spaces of polynomials such that the total degree is below a given value. Their method performs very well for dimensions $Q \geq 8$ and the integrand does not need to be periodic. Using variance reduction tools [12–14] based on approximations, one can also build Monte Carlo methods with an increased rate of convergence [15–17,23]. Based on the use of the control variates method on piecewise interpolation polynomials, Atanassov and Dimov [1] built a numerical method reaching an optimal rate of convergence for multivariate smooth functions belonging to a space where the total degree of differentiation is fixed.

We have developed in [16] a sequential Monte Carlo [8] algorithm to compute the approximation of functions on any orthogonal basis and introduced in [17] polynomial spaces very similar to Korobov ones. The use of Tchebychef polynomials combined with random drawings associated to the Tchebychef weight functions has enable to make the algorithm both fast and accurate. In order to even improve the convergence and to diminish the number of points used in the algorithm, we have proposed in [18] two new versions of the algorithm using quasi-random sequences. In this paper, we replace the all algorithm by the simple least square problem [5] of fitting our Tchebychef polynomial approximation model to some random or quasi-random points. We first make a short discussion about the way to optimize the data of a least square problem by looking at the condition number of its relative matrix A . We observe that our least square problem has a very good behaviour as its condition number goes to 1 with the number of points. Each of the coefficients of the approximation model and the value of $\int_{[-1,1]^Q} f(x) dx$ can be computed accurate up to the truncation error by solving a linear system. In order to build quadrature formulas and to avoid this resolution, we need to make the numerical computation of the inverse least square matrix. We show that the use of quasi-random sequences and especially sequences built from a quantization problem provides the most accurate quadratures for a given number of points. We compare

our quadratures to standard quadrature formulas on several numerical examples. The numerical results shows that we can obtain a better accuracy than product rules when $Q \geq 3$ and to lattices rules in any dimension. The formulas are especially efficient for very smooth functions.

2. Approximations on Tchebychef polynomials

2.1. Univariate approximation

Tchebychef polynomials $T_n(x) = \cos(n \arccos(x))$ are the orthogonal polynomials with respect to the inner product $\langle P, Q \rangle = \int_{-1}^1 \frac{P(x)Q(x)}{\sqrt{1-x^2}} dx$. They verify $\|T_0\|_2^2 = \pi$, $\|T_n\|_2^2 = \frac{\pi}{2}$ if $n \geq 1$ and also the differential equation

$$\frac{d}{dx}(\sqrt{1-x^2} T_n'(x)) + n^2 \frac{T_n(x)}{\sqrt{1-x^2}} = 0.$$

Using this equation, one can show that if $f \in C^{2L}([-1, 1])$ the coefficients

$$b_n = \frac{\langle f, T_n \rangle}{\|T_n\|_2^2} = \frac{\int_{-1}^1 \frac{f(x)T_n(x)}{\sqrt{1-x^2}} dx}{\|T_n\|_2^2}$$

of its mean-square approximation on the Tchebychef polynomials verify $|b_n| \leq \frac{C_1}{n^{2L}}$, where C_1 is a constant depending on f and L . Moreover, if we replace the approximation $\pi_N(u)$ on the polynomials of degree $\leq N$ by the interpolation polynomial $P_N(u)$ of the function u at the Tchebychef abscissae

$$y_k = \cos\left(\frac{2k+1}{N+1} \frac{\pi}{2}\right), \quad k = 0, 1, \dots, N,$$

the approximation is still as good [4]. The expression of $P_N(u)$ is given by [5]

$$P_N(u) = \sum_{n=0}^N \alpha_n T_n(x),$$

where

$$\alpha_n = \frac{\pi}{\|T_n\|_2^2 (N+1)} \sum_{k=0}^N u(y_k) T_n(y_k).$$

As $b_n = \frac{E(\pi f(V) T_n(V))}{\|T_n\|_2^2}$ where the density of V is $\frac{1}{\pi\sqrt{1-v^2}} 1_{[-1,1]}(v)$, its Monte Carlo approximation writes

$$b_n \simeq \frac{\pi}{M \|T_n\|_2^2} \sum_{i=1}^M f(X_i) T_n(V_i),$$

where the V_i s are N independent copies of V . The simulation of V can be done using the standard method by $V = \sin(\pi(U - \frac{1}{2}))$ where U is uniform on $[0, 1]$. Using a sequential Monte Carlo algorithm [16]

based on these approximations, we have computed simultaneously all the coefficients b_p of the truncated approximation

$$f_k(x) \simeq \sum_{p=1}^k b_p T_p(x)$$

up to an accuracy equal to the truncation error. The convergence of the algorithm was geometric and mainly depending on the supremum of the integrals

$$\left| \int_{-1}^1 \frac{T_l(x)T_m(x)T_i(x)T_j(x)}{\sqrt{1-x}} dx \right|,$$

where $l, m, i, j \leq k$. The main reason to use Tchebychef polynomials instead of Legendre polynomials is that this supremum is small and does not depend on k . This will be also crucial in the least square method developed in the next sections.

2.2. Multivariate approximations

We assume that f belongs to $C^{2L}([-1, 1]^Q)$. The usual method to approximate a Q -dimensional smooth function on Tchebychef polynomial bases is to use a tensor product approximation

$$f(x_1, x_2, \dots, x_Q) \simeq \sum_{i_1=0}^N \sum_{i_2=0}^N \dots \sum_{i_Q=0}^N b_{i_1, i_2, \dots, i_Q} T_{i_1}(x_1) T_{i_2}(x_2) \dots T_{i_Q}(x_Q),$$

where

$$b_{i_1, i_2, \dots, i_Q} = \frac{\langle f, T_{i_1} T_{i_2} \dots T_{i_Q} \rangle}{\prod_{j=1}^Q \|T_{i_j}\|_2^2}.$$

As in dimension one, we can replace this approximation by the interpolation polynomial $P_N(f)$ at the points of the Tchebychef grid writing

$$P_N(f) = \sum_{i_1=0}^N \sum_{i_2=0}^N \dots \sum_{i_Q=0}^N \alpha_{i_1, i_2, \dots, i_Q} T_{i_1}(x_1) T_{i_2}(x_2) \dots T_{i_Q}(x_Q),$$

where the $\alpha_{i_1, i_2, \dots, i_Q}$ are defined by

$$\alpha_{i_1, i_2, \dots, i_Q} = \frac{\pi^Q}{\prod_{j=1}^Q \|T_{i_j}\|_2^2 (N+1)^Q} \sum_{j_1=0}^N \dots \sum_{j_Q=0}^N f(y_{j_1}, \dots, y_{j_Q}) T_{i_1}(y_{j_1}) \dots T_{i_Q}(y_{j_Q}).$$

This kind of approximation is very sensitive to the dimensional effect as its complexity is a $O(N^Q)$. To attenuate this effect, one has to make selection among the basis functions to keep in the approximation and a way to compute their coefficients. Based on Smoliak’s construction, Novak and Ritter [20] have succeeded in both tasks by building quadrature formulas that are exact on spaces of multidimensional polynomials such that the total degree is below a given value. These quadrature formulas are very efficient in various situations but they are not directly linked to the smoothness of the integrand. The usual way to

take into account this smoothness is to compute multidimensional Fourier approximations on Korobov spaces which consists in keeping the basis functions such that the product of the Fourier indexes is small. Indeed if f is periodic in all the Q variables, one can easily show that the coefficients a_m of the multivariate Fourier approximation verify

$$|a_{m_1, m_2, \dots, m_Q}| \leq \frac{C}{(\tilde{m}_1 \tilde{m}_2 \cdots \tilde{m}_Q)^{2L}}$$

where C is a positive constant and with $\tilde{m} = \max(1, |m|)$. When f is not periodic, one has to use periodisation tools [2,10,13] to transform the integrand into a function which coefficients satisfy the same kind of decay. This periodisation has nevertheless bad effects on the constant C which grows very fast with the degree of periodisation and with the dimension Q . The next step is to find good lattice points [11] to annihilate the most significant Fourier coefficients that are the ones for which $\tilde{m}_1 \tilde{m}_2 \cdots \tilde{m}_Q$ is small. Letting $\hat{m} = \max(1, m)$, we have proved in [17] that

$$|b_m| \leq \frac{C_1}{(\hat{m}_1 \hat{m}_2 \cdots \hat{m}_Q)^{2L}}$$

using the differential equation satisfied by the T_n for the Q integration variables. We can then give the approximation

$$f(x_1, x_2, \dots, x_Q) \simeq \sum_{m \in W_{Q,d}} b_m T_{m_1}(x_1) T_{m_2}(x_2) \cdots T_{m_Q}(x_Q)$$

where the set

$$W_{Q,d} = \{m \in \mathbb{N}^Q / (\hat{m}_1 \cdots \hat{m}_Q) \leq d\}$$

corresponds to a level d of approximation. We give in Table 1 some values of $L_{Q,d} = \text{card}(W_{Q,d})$ to give an idea of the complexity of the approximation. Some theoretical results are described in [17].

To compute accurately the $L_{Q,d}$ coefficients belonging to this approximation set, we have used the same sequential Monte Carlo algorithm based on control variates and independent drawings of the density

$$\prod_{i=1}^Q \frac{1}{\pi \sqrt{1-x_i^2}} 1_{[-1,1]}(x_i).$$

In order to accelerate the convergence and to diminish the number of drawings, we have proposed in [18] two new versions of the algorithm based on quasi-random simulations of the previous density. The convergence of the first version was twice faster and the second version required only $5L_{Q,d}$ values to make

Table 1

d	$L_{2,d}$	$l_{3,d}$	$L_{4,d}$	$L_{5,d}$	$L_{6,d}$
1	4	8	16	32	64
2	8	20	48	112	256
3	12	32	80	192	448
5	21	62	168	432	1072
7	31	98	280	752	1936
10	48	165	504	1432	3872
15	76	276	880	2592	7232

the algorithm work. This second version was of less practical use because of its very slow speed of convergence. It has nevertheless showed that the computation of the coefficients b_m was possible using $5L_{Q,d}$ points. To make a global use of the information given by some data points to approximate the coefficients of a model, the most popular and efficient method is the least square method. We now replace our algorithm by a least square problem and discuss of its optimization.

3. Optimization of least squares approximations

3.1. Least square approximations

A least square problem [5] consists in fitting the coefficients \tilde{a}_k of an approximation model

$$f(x) \simeq \sum_{k=0}^{N-1} \tilde{a}_k \psi_k(x)$$

on some functions ψ_k knowing the values $f(X_i)$ at M points X_i of the function f to be approximated. This least square problem leads to the minimization of the functional

$$J = \frac{1}{M} \sum_{i=1}^M \left(\sum_{k=0}^{N-1} \tilde{a}_k \psi_k(X_i) - f(X_i) \right)^2$$

which is equivalent after derivation with respect to the \tilde{a}_k to the resolution of the linear system $B\tilde{a} = g$ with

$$B_{kj} = \frac{1}{M} \sum_{i=1}^M \psi_k(X_i) \psi_j(X_i), \quad g_k = \frac{1}{M} \sum_{i=1}^M \psi_k(X_i) f(X_i).$$

Of course, the function f is usually not in the approximation space defined by the functions ψ_k and its approximation writes

$$f(x) = \sum_{k=0}^{N-1} a_k \psi_k(x) + r(x),$$

where the function r measures the approximation error. The relative least square problem consists in the minimization of the functional

$$J_1 = \frac{1}{M} \sum_{i=1}^M \left(\sum_{k=0}^{N-1} a_k \psi_k(X_i) - f(X_i) + r(X_i) \right)^2.$$

This leads to a new linear system $Ba = h$ with

$$h_k = \frac{1}{M} \sum_{i=1}^M \psi_k(X_i) (f(X_i) - r(X_i))$$

which can also be written $Ba = g - \delta g$ with

$$\delta g_k = \frac{1}{M} \sum_{i=1}^M \psi_k(X_i) r(X_i).$$

Using the inequalities $\|g\| \leq \|B\| \|a\|$ and $\|a - \tilde{a}\| \leq \|B^{-1}\| \|\delta g\|$, we have

$$\|a - \tilde{a}\| \leq \|B\| \|B^{-1}\| \frac{\|\delta g\|}{\|g\|} \|a\|.$$

This shows that the upper bound on the error $\|a - \tilde{a}\|$ on the coefficients depends on two main quantities. The first one is the condition number $\|B\| \|B^{-1}\|$ of the matrix B and the second one is the relative error $\frac{\|\delta g\|}{\|g\|}$ on the approximation model which depends on $\|r\|$. We now try to optimize the choice of the data points and of the model to minimize the previous quantities.

3.2. Optimization

We have $\|B\| \|B^{-1}\| \geq 1$ for any matrix B and $\|B\| \|B^{-1}\| = 1$ if and only if $B = \alpha I$. Hence, the ideal situation in terms of minimization is that the matrix B is equal or at least close to αI . A classic method to realize this ideal situation on an interval is to use an approximation on orthonormal polynomials P_k with respect to the inner product $\langle u, v \rangle = \int_a^b v(x)u(x)q(x) dx$, where q is a positive weight function. This approximation writes

$$P_N(f) \simeq \sum_{k=0}^N \alpha_k P_k$$

with $\alpha_k = \langle f, P_k \rangle$ and the points X_i are used to build quadrature formulas to compute accurately the coefficients α_k . These points are usually chosen as the zeros of P_{N+1} which makes the quadrature formula exact for all polynomials of degree $\leq 2N + 1$. As in this case all the coefficients of the least-square matrix are equal to a numerical computation of $\langle P_j, P_k \rangle$ using this quadrature formula, we are in this ideal situation. For example, the approximation on Tchebychef polynomials writes $P_N(f) = \sum_{n=0}^N \alpha_n T_n$ with

$$\alpha_n = \frac{\pi}{\|T_n\|_2^2 (N+1)} \sum_{k=0}^N f(y_k) T_n(y_k).$$

In higher dimensions, one can still have the same situation by using tensor product approximations on multidimensional orthogonal polynomials. As we mentioned before, the problem of this kind of approximation is its complexity.

To decrease the complexity of the approximations on Tchebychef orthogonal polynomials, we have described in Section 2 an approximation on reduced size bases. In order to compute the coefficients of this approximation, we now use random numbers in the least square approximation. We first describe the situation in the one-dimensional case. The coefficients

$$B_{kj} = \frac{1}{M} \sum_{i=1}^M T_k(X_i) T_j(X_i)$$

should appear as a Monte Carlo computation of

$$\langle T_k, T_j \rangle = \int_{-1}^1 \frac{T_k(x) T_j(x)}{\sqrt{1-x^2}} dx.$$

As the X_i are chosen independent with density $\frac{1}{\pi\sqrt{1-x^2}}1_{[-1,1]}(x)$, we have

$$\int_{-1}^1 \frac{T_k(x)T_j(x)}{\sqrt{1-x^2}} dx = E(\pi T_k(X)T_j(X)) \simeq \frac{\pi}{M} \sum_{i=1}^M T_k(X_i)T_j(X_i).$$

We have hence

$$E(B_{kj}) = \frac{1}{\pi} \int_{-1}^1 \frac{T_k(x)T_j(x)}{\sqrt{1-x^2}} dx = \delta_{kj}, \quad \text{Var}(B_{kj}) = \frac{1}{\pi M} \int_{-1}^1 \frac{T_k^2(x)T_j^2(x)}{\sqrt{1-x^2}} dx.$$

As $|T_k| \leq 1$, the variance of B_{kj} is bounded by $\frac{1}{M}$. This means that the variance does not increase with the number of basis functions. This would not be true, for example, with the use of Legendre polynomials [16]. Hence the choice of the Tchebychef polynomials as the approximation basis is really important. In higher dimensions, the points are drawn according to the density $\prod_{i=1}^Q \frac{1}{\pi\sqrt{1-x_i^2}}1_{[-1,1]}(x_i)$ and we still have such a nice speed of convergence for the coefficients of the least square matrix. In Section 5 we will try to fasten the convergence of this matrix to the identity matrix by replacing the random drawings by quasi-random points.

4. A least square method for multivariate integration

4.1. The approximation model

The approximation of the function f writes

$$f(x_1, x_2, \dots, x_Q) \simeq \sum_{m \in W_{Q,d}} b_m T_{m_1}(x_1) T_{m_2}(x_2) \cdots T_{m_Q}(x_Q).$$

We compute the $L_{Q,d}$ coefficients b_k belonging to $W_{Q,d}$ using a program which tests if $m_1 m_2 \cdots m_Q \leq d$ and stores the values of these parameters in Q lists $l_1(k), l_2(k), \dots, l_Q(k)$. We also define $c_1(k) = 1_{l_1(k) \geq 1}, \dots, c_Q(k) = 1_{l_Q(k) \geq 1}$ which will occur in the following normalizations. We now write

$$f(x_1, x_2, \dots, x_Q) \simeq \sum_{k=1}^{L_{Q,d}} b_k T_{l_1(k)}(x_1) T_{l_2(k)}(x_2) \cdots T_{l_Q(k)}(x_Q).$$

4.2. The least square problem

We could now solve the least square problem by minimizing

$$J = \frac{1}{M} \sum_{i=1}^M \left(\sum_{k=1}^{L_{Q,d}} \tilde{b}_k \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) - f(X_1^{(i)}, \dots, X_Q^{(i)}) \right)^2.$$

In order to obtain a least square matrix which goes to identity with M , we prefer to write a new weighted least square problem

$$J_1 = \frac{1}{M} \sum_{i=1}^M \left(\sum_{k=1}^{L_{Q,d}} \tilde{d}_k \sqrt{2^{\sum_{n=1}^Q c_n(k)}} \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) - f(X_1^{(i)}, \dots, X_Q^{(i)}) \right)^2$$

with

$$\tilde{d}_k = \frac{\tilde{b}_k}{\sqrt{2^{\sum_{n=1}^Q c_n(k)}}}.$$

The minimization of J_1 is equivalent to the resolution of the system $B\tilde{d} = q$ with

$$B_{k,j} = \frac{\sqrt{2^{\sum_{n=1}^Q c_n(k)+c_n(j)}}}{M} \sum_{i=1}^M \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) T_{l_n(j)}(X_n^{(i)})$$

and

$$q_k = \frac{\sqrt{2^{\sum_{n=1}^Q c_n(k)}}}{M} \sum_{i=1}^M \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \dots, X_Q^{(i)}).$$

4.3. Numerical integration

4.3.1. Direct integration

Once the coefficients \tilde{b}_k have been computed, we obtain the approximation

$$f(x_1, x_2, \dots, x_Q) \simeq \sum_{k=1}^{L_{Q,d}} \tilde{b}_k T_{l_1(k)}(x_1) T_{l_2(k)}(x_2) \cdots T_{l_Q(k)}(x_Q).$$

As we are in general concerned by the computation of an approximation $\tilde{I}(f)$ of

$$I(f) = \int_{[-1,1]^Q} f(x) dx,$$

we just write

$$\tilde{I}(f) = \sum_{k=1}^{L_{Q,d}} \tilde{b}_k \prod_{n=1}^Q \int_{-1}^1 P_{l_n(k)}(x) dx.$$

4.3.2. Quadrature formulas

To build quadrature formulas, we first compute numerically the inverse matrix B^{-1} and we then write $\tilde{d} = B^{-1}q$ to obtain

$$\tilde{d}_k = \sum_{j=1}^{L_{Q,d}} B_{kj}^{-1} q_j = \sum_{j=1}^{L_{Q,d}} B_{kj}^{-1} \frac{\sqrt{2^{\sum_{n=1}^Q c_n(j)}}}{M} \sum_{i=1}^M \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \dots, X_Q^{(i)})$$

that is

$$\tilde{d}_k = \sum_{i=1}^M \left(\sum_{j=1}^{L_{Q,d}} B_{jk}^{-1} \frac{\sqrt{2^{\sum_{n=1}^Q c_n(j)}}}{M} \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \dots, X_Q^{(i)}) \right).$$

Then, we have

$$\tilde{b}_k = \sum_{i=1}^M \beta_{i,k} f(X_1^{(i)}, \dots, X_Q^{(i)})$$

with

$$\beta_{i,k} = \sqrt{2^{\sum_{n=1}^Q c_n(k)}} \sum_{j=1}^{L_{Q,d}} B_{jk}^{-1} \frac{\sqrt{2^{\sum_{n=1}^Q c_n(j)}}}{M} \prod_{n=1}^Q T_{l_n(k)}(X_n^{(i)}).$$

We finally have

$$\tilde{I}(f) = \sum_{k=1}^{L_{Q,d}} \tilde{b}_k \prod_{n=1}^Q \int_{-1}^1 P_{l_n(k)}(x) dx = \sum_{i=1}^M \alpha_i f(X_1^{(i)}, \dots, X_Q^{(i)})$$

with

$$\alpha_i = \sum_{k=1}^{L_{Q,d}} \beta_{i,k} \prod_{n=1}^Q \int_{-1}^1 P_{l_n(k)}(x) dx.$$

We have observed no significant difference between the two numerical integration procedures on some numerical tests. We hence use the quadrature formulas in all the numerical examples as they are cheaper in terms of computation. In some of the numerical examples, we will also need to compute integrals on $[0, 1]^Q$. The relative quadrature formula simply writes

$$\sum_{i=1}^M \frac{\alpha_i}{2^Q} f\left(\frac{X_1^{(i)} + 1}{2}, \dots, \frac{X_M^{(i)} + 1}{2}\right).$$

4.4. Resolution of the linear systems

We have to solve linear systems $B\tilde{d} = q$ where the matrix B is symmetric definite positive and very close to the identity matrix. The conjugate gradient method is very efficient in this case as the condition number is small and that q is a very good initial guess for the solution. We have nevertheless chosen to use the Cholesky decomposition $B = LL^t$ as we have to compute B^{-1} numerically to build the quadrature formulas. Indeed, this computation induces many resolutions of linear systems with the same matrix B . For the moment, we have limited ourselves to matrices which sizes are less than 3000 for practical reasons (the use of Matlab in the matrix computations). Of course, we need to perform this computation for larger matrices if we wish to build efficient formulas in rather large dimensions. This computation can be very expensive for huge matrices but anyway the quadratures formulas are built once and for all.

5. Optimization of the data points

5.1. Statement of the problem

The aim of this section is to minimize the number M of data points to achieve a good approximation of our least square problem. This approximation depends on the condition number of the least square matrix B . The faster the coefficients $B_{k,j}$ go to δ_{kj} with M , the closer to 1 this condition number is. As these coefficients are integrals computed by means of a Monte Carlo method, the rate of convergence is a $O(\frac{1}{\sqrt{N}})$. As mentioned in the introduction, it can be efficient to replace this Monte Carlo approximation by an approximation using quasi-Monte Carlo sequences as their rate of convergence is a $O(\frac{\ln(N)^{Q-1}}{N})$. Another similar point of view is to find the best way to represent with M points the density

$$w(x) = \prod_{i=1}^Q \frac{1}{\pi \sqrt{1-x_i^2}} 1_{[-1,1]}(x_i)$$

according to some criterion. We are going to use 4 different ways to build our data points and compare the accuracies obtained on some numerical examples. We use a pseudo-random linear generator, Halton sequences, Sobol sequences and points build from a quantization problem.

5.2. Quantization

We give some more details about quantization which is not so standard for numerical integration than Halton or Sobol sequences. Optimal quantization in the quadratic case consists in finding the M points in $D = [-1, 1]^Q$ minimizing the functional

$$J(M) = \min \left(\int_D \inf_{1 \leq i \leq M} |x - x_i|^2 w(x) dx : \{x_1, x_2, \dots, x_M \in D\} \right).$$

We use this quadratic criterion as we want to solve this problem numerically and that in this case there is an efficient algorithm to do it. This competitive learning vector quantization algorithm (see [3]) can be shortly described as follows. First draw M independent points X_i according to the density w . Draw one more point Y_1 from the same density and find the closest point X_{\min} to Y_1 among the X_i . Move a little X_{\min} towards Y_1 such that its new location is defined by

$$X_{\min} + \varepsilon(Y_1 - X_{\min})$$

where $\varepsilon > 0$ is a small parameter. The point Y_1 is then removed, another point Y_2 is drawn and so on. The parameter ε decreases with the number n of steps of the algorithm. A very complete discussion on the numerical aspects of the algorithm is done in [21]. We use here a version of the algorithm where the value of the parameter ε at step i is $\varepsilon(i) = \varepsilon_1 + \frac{(\varepsilon_2 - \varepsilon_1)i}{n}$. We take $\varepsilon_1 = 10^{-2}$, ε_2 either equal to 10^{-4} or 10^{-5} and n either equal to 2×10^7 or 2×10^8 in the numerical experiments. It was first proposed in [22] to use the points x_i corresponding to the optimal quadratic quantization for numerical integration. For each point x_i a tessell

$$C_i = \{u \in D \mid \|x_i - u\| < \|x_k - u\|, k \neq i\}$$

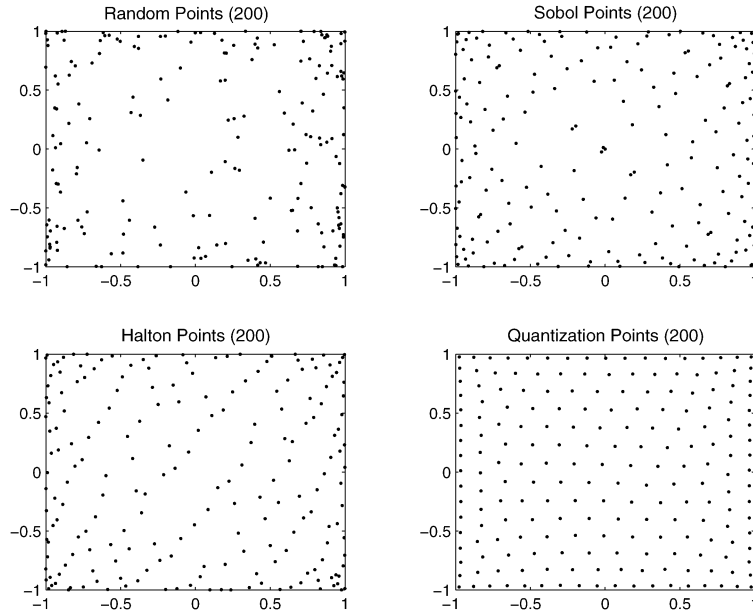


Fig. 1.

is associated. Then the approximation of $\int_D f(x)w(x) dx$ is given by the quadrature formula

$$\sum_{i=1}^n \left(\int_{C_i} w(x) dx \right) f(x_i).$$

These formulas can take into account some usual kinds of regularity of the function f and are more accurate than Monte Carlo integration in rather low dimensions.

5.3. Numerical results

We first plot on a bidimensional domain 200 data points using each sequence of points to have a geometric idea of the location of these points (see Fig. 1).

More points are located near the boundaries because the density function w takes larger values there. We observe that the quantization method provides the better-looking distribution. In order to give a more quantitative conclusion, we now check the accuracy of the relative quadrature formulas on the example of the function $f(x, y, z) = \exp(\frac{x+y+z}{3})$ on $[-1, 1]^3$.

Given a value $M = 310$ of the number of data points, we plot the accuracy as a function of the ratio $M/L_{Q,d}$.

The quantization points provide the more accurate quadrature formulas for all the values of the ratio $M/L_{Q,d}$ plotted in Fig. 2. Nevertheless beyond a ratio equal to 2 and especially when this ratio is close to 1, the behaviour of the quadratures becomes quite erratic. In some examples that we have studied, the Halton sequences or even the random points can provide better results than the quantization points in this area. For the sake of stability, we choose $M/L_{Q,d} = 3$ for which the same behaviour was observed on all the numerical tests. In this case, the quantization quadratures are about 3 to 10 times more accurate than

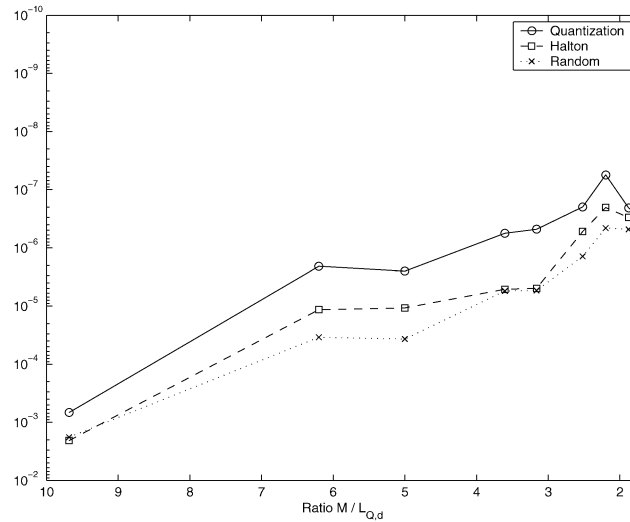


Fig. 2.

the Halton quadratures and at least 10 times more accurate than the random quadratures. An additional nice property is that all the weights of the quadrature formulas built from the quantization points are positive for such a value of the ratio. However these quantization points can be difficult to build in practice especially where the number of points becomes large. Hence, we now use Halton sequences in all the following numerical results even though the relative quadrature formulas are not quite as accurate as the ones built from the quantization points.

6. Numerical results

6.1. Tests on very smooth functions

We first take the example of the integral

$$I = \int_{[0,1]^6} \exp\left(\frac{x+y+z+t+u+v}{6}\right) dx dy dz dt du dv$$

which has been studied, for example, in [13, p. 224]. This function is obviously very smooth and hence well adapted to our quadratures. The value of the integral I is $(6(\exp(\frac{1}{6}) - 1))^6 \simeq 1.66$. We study in Table 2 the absolute error $|I - \tilde{I}|$ as a function of the number $M = 3L_{6,d}$ of integration points.

We achieve an accuracy up to 12 digits with 8304 integration points. In order to compare with lattice rules, we now transform the original integrand using the polynomial transformation $P(t) = 3t^2 - 2t^3$ for periodisation in each of the integration variables. The new integral to compute is now

$$\int_{[0,1]^6} f_1(P(x_1), \dots, P(v_1)) P'(x_1) \cdots P'(v_1) dx_1 \cdots dv_1.$$

Table 2

d	$L_{6,d}$	M	$ I - \tilde{I} $
2	256	768	6×10^{-6}
3	448	1344	8×10^{-7}
5	1072	3216	4×10^{-9}
8	2768	8304	1×10^{-12}

Table 3

d	$L_{4,d}$	M	$ J - \tilde{J} $
1	16	48	2×10^{-2}
3	80	240	5×10^{-5}
6	248	744	7×10^{-6}
17	1041	3123	4×10^{-8}
30	2453	7359	4×10^{-10}

Using our quadratures on this transformed integral, we achieve an accuracy of 2 digits when using 8304 points. The results taken from [13] using lattice rules give an accuracy of 3 digits when using 10^4 points and 4 digits with 4×10^4 points. These results clearly show the effect of periodisation on the function f . The new integrand f_1 can no more be well approximated on the polynomial basis because of its quicker variations. The lattice rules seem slightly more accurate on the transformed integral but they hardly provide any error reduction on the original nonperiodic integral [13]. Hence for the same number of points ($\simeq 10^4$), we achieve an accuracy of 12 digits with our quadratures and of 3 digits with lattice rules. Some more examples developed in [17] have confirmed the bad effects of periodisation. As a conclusion, we think that our quadratures are preferable to lattice rules except maybe if the original integrand is periodic or when periodisation removes a singularity in the integrand.

We now consider the integral

$$J = \int_{[0,1]^4} \exp(x) \sin(y) \cos(z) \log(1+t) \, dx \, dy \, dz \, dt$$

which has already being studied in [1,17]. We give in Table 3 the absolute error $|J - \tilde{J}|$ as a function of the number $M = 3L_{4,d}$ of integration points.

The method developed in [1] attains an optimal rate of convergence of $\frac{1}{M^{1/2+k/Q}}$ for C^k_Q real functions that is for functions possessing all the partial derivatives

$$\frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \dots \partial x_1^{\alpha_Q}}, \quad \alpha_1 + \dots + \alpha_Q = r$$

continuous when $r < k$ and bounded in sup norm when $r = k$. It is based on the use of the control variates method combined with piecewise polynomial interpolations at $\frac{(Q+k-1)!}{Q!(k-1)!}$ well chosen deterministic points on each of n^Q cubes. This method requires obviously less regularity than ours. Nevertheless it has been tested on J where the integrand is C^∞ with $Q = k = 4$ and $n = 10$. The accuracy was 10^{-8} with about 4×10^5 integration points which was already comparable to our previous algorithm [17]. This corresponds to a numerical approximation of $\frac{-\log(10^{-8})}{\log(4 \times 10^5)} \simeq 1.4$ of the order of the method. The numerical approximation of the order of the new quadratures is $\frac{-\log(4 \times 10^{-10})}{\log(7359)} \simeq 2.4$ which is a lot better. Moreover, the CPU times are drastically reduced as we now have quadrature formulas instead of an algorithm.

6.2. Tests on less smooth functions

We have for the moment made some numerical tests on very smooth functions which appears quite satisfying. In the paper from Hickernel [9] is mentioned some properties that should be verified by a

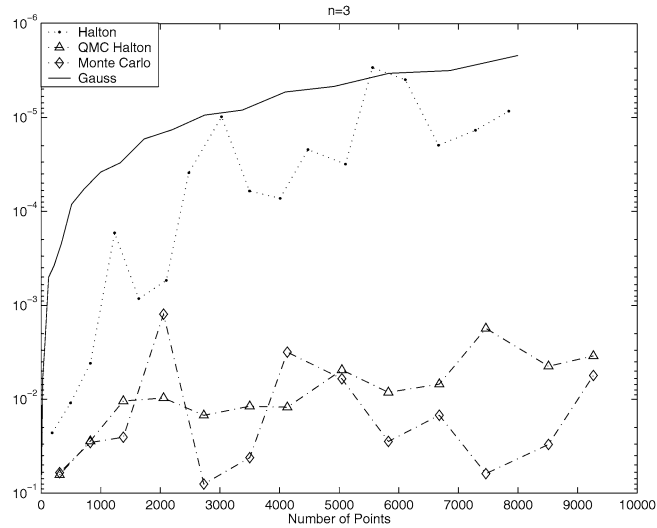


Fig. 3.

“dream” quadrature rule. One of the requirements is that this quadrature should not only integrate accurately very smooth functions but also being efficient for functions with less regularity. To check if our quadrature possesses this good property, we now make some comparison with crude Monte Carlo and quasi-Monte Carlo on the integrals

$$\int_{[-1,1]^3} g_n(x)g_n(y)g_n(z) \, dx \, dy \, dz$$

where

$$g_n(x) = -x^n \exp(x)1_{[-1,0]}(x) + x^n \cos(x)1_{[0,1]}(x).$$

We intend moreover to show that our quadratures can reach a better accuracy than product rules even for low dimensions. Hence, we also compute the integral of the tensored Tchebycheff interpolation polynomial at the Gauss grid defined in Section 2.2 and compare its value with our quadratures. We first plot the accuracy as a function of the number of integration points for $n = 3$ in the logarithm scale. (See Fig. 3.)

We can see on this example that our quadratures are slightly less accurate than the Gauss-type formulas. We achieve for example an accuracy of roughly 5 digits using 8000 quadrature points for each of these two integration methods. We can expect that the use of quantization points would have made these two quadratures equivalent on this example. If we now compare to the usual Monte Carlo or quasi-Monte Carlo methods, we see that they can hardly reach an accuracy of 2 or 3 digits. Our quadratures have taken advantage of the smoothness of the function. We now plot the results on a even less smooth function that is for $n = 1$. (See Fig. 4.)

The situation is quite different than in the previous example. Our quadratures are the most accurate just before the quasi-Monte Carlo sequences. Gauss-type formulas and the Monte Carlo method give roughly the same results and are really worse than the others. The integrand is only continuous in this example which explains why Gauss-type formulas do not work well. The reason why our quadratures are

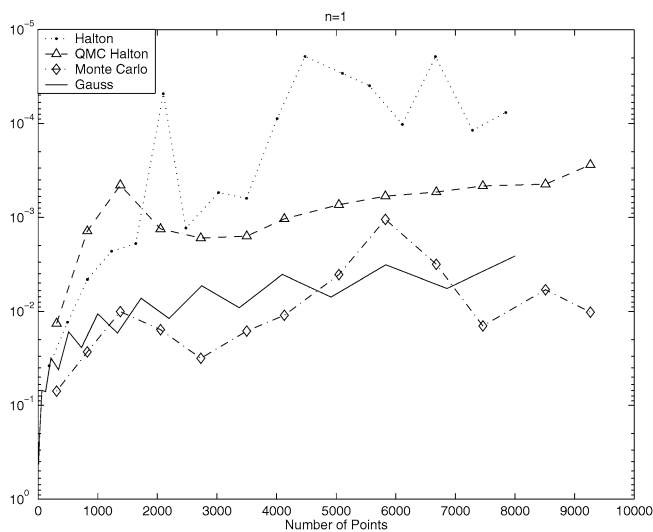


Fig. 4.

still accurate is that they have a quasi-Monte Carlo structure and keep at least the rate of convergence of the quasi-Monte Carlo sequences when the function is not smooth.

7. Conclusion

We have built quadrature formulas for the numerical integration of multivariate smooth functions by fitting a Korobov-type multivariate Tchebychef polynomial approximation to quasi-random values. The same quadrature formula can be really efficient for very smooth functions but also work well for less regular integrands. The use of points build from optimal quadratic quantization appears as the most efficient choice in the quadrature points. It provides more accurate quadratures with positive weights in the studied examples. However, building these quantization points requires a lot of computational work for a large number of points in higher dimensions. Another difficult task is the numerical inversion of large linear systems when building quadratures in higher dimensions than the ones considered here.

We have especially focused on numerical integration but many applications of the polynomial approximation on a reduced size basis are possible. We can mention Fredholm integral equations or spectral methods for partial differential equations. In particular, coupling this approximation and the sequential Monte Carlo algorithm developed in [7] seems really promising for the numerical solution of partial differential equations. These ideas are under development.

References

- [1] E.I. Atanassov, I.T. Dimov, A new optimal Monte Carlo method for calculating integral of smooth functions, *Monte Carlo Methods Appl.* 5 (2) (1999) 149–167.
- [2] M. Beckers, A. Haegmans, Transformations of integrands for lattice rules, in: T.O. Espelid, A. Genz (Eds.), *Numerical Integration-Recent Developments, Software and Applications*, Kluwer Academic, Dordrecht, 1992, pp. 329–340.

- [3] A. Benveniste, M. Metivier, P. Priouret, *Algorithmes Stochastiques et Approximations Stochastiques*, Masson, Paris, 1987.
- [4] C. Bernardi, Y. Maday, *Approximations Spectrales de Problèmes aux Limites Elliptiques*, Springer, Berlin, 1992.
- [5] A. Bjork, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.
- [6] P. Davis, P. Rabinowitz, *Methods of Numerical Integration*, second ed., *Comput. Sci. Appl. Math.*, Academic Press, New York, 1984.
- [7] E. Gobet, S. Maire, A spectral Monte Carlo method for the Poisson equation, *Monte Carlo Methods Appl.* 10 (3–4) (2004) 275–285.
- [8] J. Halton, Sequential Monte Carlo, *Proc. Cambridge Philos. Soc.* 58 (1962) 57–78.
- [9] F.J. Hickernel, My dream quadrature rule, *J. Complexity* 19 (2003) 420–427.
- [10] M. Iri, S. Moriguti, Y. Takazawa, On a certain quadrature formula, *Kokyuroku Res. Inst. Math. Sci. Kyoto Univ.* 91 (1970) 82–118 (in Japanese).
- [11] S. Joe, I. Sloan, Imbedded lattice rules for multidimensional integration, *SIAM J. Numer. Anal.* 29 (4) (1992) 1119–1135.
- [12] M.H. Kalos, P.A. Whitlock, *Monte Carlo Methods*, Wiley, New York, 1986.
- [13] A.R. Krommer, C.W. Ueberhuber, *Computational Integration*, SIAM, Philadelphia, PA, 1998.
- [14] B. Lapeyre, E. Pardoux, R. Sentis, *Méthodes de Monte-Carlo pour les Équations de Transport et de Diffusion*, Springer, Berlin, 1998.
- [15] G.P. Lepage, A new algorithm for adaptive multidimensional integration, *J. Comput. Phys.* 27 (1978) 192–203.
- [16] S. Maire, Reducing variance using iterated control variates, *J. Statist. Comput. Simulation* 73 (1) (2003) 1–29.
- [17] S. Maire, An iterative computation of approximations on Korobov-like spaces, *J. Comput. Appl. Math.* 157 (2003) 261–281.
- [18] S. Maire, Polynomial approximations of multivariate smooth functions from quasi-random data, *Statist. Comput.* 14 (2004) 333–336.
- [19] H. Niederreiter, Quasi-Monte Carlo methods and pseudorandom numbers, *Bull. Amer. Math. Soc.* 84 (1978) 957–1041.
- [20] E. Novak, K. Ritter, High dimensional integration of smooth functions over cubes, *Numer. Math.* 75 (1996) 79–97.
- [21] G. Pages, A space vector quantization for numerical integration, *J. Comput. Appl. Math.* 89 (1997) 1–38.
- [22] G. Pages, J. Printems, Optimal quadratic quantization for numerics: the Gaussian case, *Monte Carlo Methods Appl.* 9 (2) (2003) 135–166.
- [23] W.H. Press, G.R. Farrar, Recursive stratified sampling for multidimensional Monte Carlo integration, *Comput. Phys.* 4 (1990) 190–195.
- [24] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in FORTRAN*, Cambridge University Press, Cambridge, 1996.
- [25] I.H. Sloan, P.J. Kachoyan, Lattice methods for multiple integration: Theory, error analysis and examples, *SIAM J. Numer. Anal.* 24 (1987) 116–128.
- [26] S.A. Smoliak, Quadrature and interpolation formulas for tensor products of certain classes of functions, *Soviet Math. Dokl.* 4 (1963) 240–243.
- [27] I.M. Sobol, The distributions of points in a cube and the approximate evaluation of integrals, *Zh. Vychisl. Mat. Mat. Fiz.* (1967) 784–802.