



ELSEVIER

# An iterative computation of approximations on Korobov-like spaces

Sylvain Maire\*

*ISITV, Université de Toulon et du Var, Avenue G. Pompidou, BP 56, 83262 La Valette du Var Cedex, France*

Received 21 August 2002; received in revised form 13 January 2003

## Abstract

This paper treats the multidimensional application of a previous iterative Monte Carlo algorithm that enables the computation of approximations in  $L^2$ . The case of regular functions is studied using a Fourier basis on periodised functions, Legendre and Tchebychef polynomial bases. The dimensional effect is reduced by computing these approximations on Korobov-like spaces. Numerical results show the efficiency of the algorithm for both approximation and numerical integration.

© 2003 Elsevier B.V. All rights reserved.

*Keywords:* Monte Carlo method; Iterative algorithm; Polynomial approximations; Korobov spaces; Numerical integration

## 1. Introduction

The Monte Carlo approximation of the integral

$$I = \int_D f(x) dx,$$

where  $D = [0, 1]^Q$  with  $Q \in \mathbb{N}^*$  and where  $f \in L^2(D)$  is given by

$$\bar{I} = \frac{1}{N} \sum_{i=1}^N f(X_i),$$

where  $X_i$  are  $N$  independent copies of a  $Q$ -dimensional random variable  $X$  whose components are uniform and independent on  $[0, 1]$ . The rate of convergence of this approximation given by the central

\* Tel.: 04-94-14-25-58; fax: 04-94-14-24-48.

E-mail address: [maire@isitv.univ-tln.fr](mailto:maire@isitv.univ-tln.fr) (S. Maire).

limit theorem is  $\sigma_f/\sqrt{N}$  where  $\sigma_f^2$  is the variance of the random variable  $f(X)$  that is

$$\sigma_f^2 = \int_D f^2(x) dx - \left( \int_D f(x) dx \right)^2.$$

This makes the Monte Carlo integration a very simple tool to achieve moderate accuracy even for high-dimensional irregular functions. A natural way to increase this accuracy is to find a function  $\tilde{f}$  such that  $\int_D \tilde{f}(x) dx = \int_D f(x) dx$  and for which  $\sigma_{\tilde{f}}^2 < \sigma_f^2$  holds. This can be done by using a wide number of methods, called variance reduction methods, among which the most important are importance sampling, antithetic variables, stratification method or control variates. For more details about these methods, one can check [10–12,18]. One can use these methods to obtain Monte Carlo methods with increased convergence rate. This is for instance done and described in [5,13,16,17]. Atanassov and Dimov [1] have built a Monte Carlo method with an optimal rate of convergence for regular multidimensional functions. Of course, these methods are very sensitive to the dimensional effect because they more or less rely on an approximation of the function  $f$ . We have developed in a previous work [14] a Monte Carlo algorithm which enables the computation of approximations into  $L^2(D)$  on any orthonormal basis. It is based on an iterative application of the control variate method and has led to Monte Carlo estimators for numerical integration with increased rate of convergence in dimension one for regular functions. Our goal here is to show how to use this algorithm for regular functions in moderate dimension in trying to attenuate the dimensional effect. This can be done if we can make a choice among the coefficients to keep in the multidimensional approximation. This is usually achieved for the  $Q$ -dimensional Fourier basis by introducing Korobov spaces [11,15] which rely on the decay of the Fourier coefficients  $a_m$  as

$$|a_m| \leq \frac{c}{(\widetilde{m}_1 \widetilde{m}_2 \dots \widetilde{m}_Q)^\alpha},$$

where  $\widetilde{m} = \sup(1, |m|)$  and  $\alpha > 1$ . To achieve such a decay for non-periodic functions, one has to transform the original integral using the periodisation method [2,8,11,19] so that the function and its derivative vanish at the extremities of the integration domain. This enables one to build integration formulas known as lattice rules [9] which try to annihilate the most significant Fourier coefficients belonging to Zaremba crosses  $Z_{Q,d}$  which are defined as

$$Z_{Q,d} = \{m \in \mathbb{Z}^Q / (\widetilde{m}_1 \dots \widetilde{m}_Q) = d\}.$$

Sloan and Kachoyan [20] give an estimation of the accuracy of these lattice rules which is mainly described by  $O(1/\rho(L)^\alpha)$  where

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

for a given integration lattice  $L$ . We will achieve the same kind of estimation using our algorithm on regular periodised functions with an additional term due to the iterative method. The main drawback of the periodisation method is an artificial effect on the constant  $c$ . It grows rapidly when  $\alpha$  increases. This is true in one dimension [7] and even more in higher dimension [4] so that  $\alpha$  is usually about 3. Approximations of regular functions on polynomial bases [3] check the same kind of properties as the Fourier basis on periodised functions with fewer terms in the approximation and no need of periodisation. We will compute these approximations on Legendre and Tchebychef polynomial

bases. The faster version of the algorithm using Tchebychef polynomials will be compared in terms of numerical integration to the method developed in [1].

## 2. Description of the approximation algorithm

We begin by a short description of the approximation algorithm and of its intrinsic properties (see [14] for more details). We want to compute on  $D = [0, 1]^Q$  the coefficients  $a_k = \langle f, e_k \rangle$  of the approximation of a function  $f$  on an orthonormal basis of  $L^2(D)$  whose basis functions  $e_k$  are bounded. We have

$$f(x) = \sum_{k=1}^p a_k e_k(x) + r(x)$$

with  $a_k = \langle f, e_k \rangle$  and  $\langle r, e_k \rangle = 0$ . Their approximations as well as  $f$ 's are obtained using  $N$  uniform and independent random drawings  $X_i$  by

$$a_k^{(1)} = \frac{1}{N} \sum_{i=1}^N f(X_i) e_k(X_i), \quad f^{(1)}(x) = \sum_{k=1}^p a_k^{(1)} e_k(x).$$

Then we compute a correction

$$b_k^{(1)} = \frac{1}{N} \sum_{i=1}^N (f(Y_i) - f^{(1)}(Y_i)) e_k(Y_i)$$

on these coefficients using this time random drawings  $Y_i$  independent of the previous ones. We obtain a new approximation of  $a_k$  and  $f$  at step two by

$$a_k^{(2)} = a_k^{(1)} + b_k^{(1)}, \quad f^{(2)}(x) = \sum_{k=1}^p a_k^{(2)} e_k(x).$$

If we now perform  $M$  steps of the previous algorithm and if we replace  $f$  by its algebraic expression, we obtain

$$a_k^{(M)} = \sum_{j=1}^p Q_{k,j}^{(M)} a_j + T_k^{(M)}, \quad f^{(M)}(x) = \sum_{k=1}^p a_k^{(M)} e_k(x).$$

The term  $\sum_{j=1}^p Q_{k,j}^{(2)} a_j$  represents the estimation of  $a_k$  if  $r(x) \equiv 0$ . The following lemma shows that it is unbiased and that the variance of its components reduces geometrically if the number of drawings used at each step of the algorithm is sufficiently large.

**Lemma 2.1.** *There are constants  $C(p)$  and  $K(p) \leq p^2$ , depending only of the approximation basis such that*

$$E(Q_{k,j}^{(M)}) = \delta_{k,j}, \quad \text{Var}(Q_{k,j}^{(M)}) \leq \frac{K(p)^{M-1}}{N^M} C(p)^M.$$

The constants  $K(p)$  and especially  $C(p)$  determine the speed of convergence of the algorithm. If we choose  $s, l, k, j \leq p$  and define

$$E_{k,j} = \sup_{s,l \neq k,j} \left| \int_D e_k(x)e_j(x)e_s(x)e_l(x) dx \right|,$$

$C(p)$  depends mainly on the maximum of the  $E_{k,j}$  and  $K(p)$  on the number of non-zero integrals intervening in the definition of the  $E_{k,j}$ . An important task is to try to compute these constants and to find approximation basis for which they are as small as possible. We also give in the following lemma a control depending on  $r(x)$  of the term  $T_k^{(2)}$  which is linked to the truncation error.

**Lemma 2.2.** *There are constants  $\gamma(p)$  and  $\gamma_1(p)$  such that*

$$\text{Var}(T_k^{(2)}) \leq 2 \left( \frac{\gamma(p)}{N} + \frac{p^2}{N^2} \gamma_1(p) \right) \int_D r^2(x) dx$$

with

$$\gamma(p) = \sup_{1 \leq k \leq p} \sup_{x \in D} e_k^2(x)$$

and

$$\gamma_1(p) \leq \gamma(p)C(p).$$

These two lemmas are valid in any dimension. Nevertheless, we will first explain how to make good use of the algorithm in dimension one for regular functions. The generalisation to higher dimensions will then come very naturally.

### 3. The monodimensional case

We will now give the performances of our algorithm assuming that the coefficients  $a_k$  decrease as  $C/k^L$  where  $L$  and  $C$  are positive constants. This is the natural framework for regular functions in dimension one. We will first use the Fourier basis on periodised functions and then Legendre and Tchebychef polynomial basis for which such an assumption holds. An accurate study of the algorithm and especially of the constants  $C(p)$  and  $K(p)$  will show the superiority of the algorithm based on Tchebychef polynomials.

#### 3.1. Polynomial decay of the $a_k$

The following theorem describes the accuracy of our estimators for each of the coefficients of the approximation basis and for the approximation itself when the  $a_k$  satisfy  $|a_k| \leq C/k^L, \forall k \geq 1$ . We will write

$$f(x) = \sum_{k=1}^p a_k e_k(x) + r(x)$$

with

$$r(x) = \sum_{k=p+1}^{\infty} a_k e_k(x)$$

and the approximation of  $f$  at the  $M$ th step will be given by

$$f^{(M)}(x) = \sum_{k=1}^p a_k^{(M)} e_k(x).$$

**Theorem 3.1.** *Assuming the previous hypotheses and that*

(i)

$$\frac{p}{N} < 1,$$

(ii)

$$\tau(p) = \max \left( \sup_{1 \leq k \leq p} \int_D (1 - e_k^2(x))^2 dx, \sup_{1 \leq j, k \leq p, j \neq k} \int_D e_k^2(x) e_j^2(x) dx \right) \leq \frac{N}{4},$$

we have

$$E(a_k^{(M)}) = a_k, \quad \text{Var}(a_k^{(M)}) \leq 2 \left( \mu(p) \frac{1}{p^{2L-1}} + \mu_1 \frac{K(p)^{M-1}}{N^M} C(p)^M \right)$$

and also

$$E \left( \int_D (f(x) - f^{(M)}(x))^2 dx \right) \leq p \text{Var}(a_k^{(M)}) + \mu_2 \frac{1}{p^{2L-1}},$$

where  $\mu(p)$ ,  $\mu_1$ ,  $\mu_2$ ,  $K(p)$  and  $C(p)$  are positive constants.

**Proof.** See [14].  $\square$

### 3.2. Fourier basis on periodised functions

The coefficients of the approximation of a function  $f$  on the Fourier basis do not usually satisfy a decay as  $|a_k| \leq C/k^L$ . Even for very regular functions, such a decay can only be achieved if  $f$  is somehow periodic. The periodisation method is the usual trick to ensure this decay and also to get rid of singularities occurring at the boundaries of the integration domain. We will describe this method for  $f \in C^\infty([0, 1])$  using only polynomial changes of variables for periodisation. We consider the polynomial  $P$  of lowest degree such that  $P(0)=0$ ,  $P(1)=1$  and  $P^{(i)}(0)=P^{(i)}(1)=0$  if  $1 \leq i \leq L$ . One can easily see that  $P$  is increasing, that the function  $g(t) = f(P(t))P'(t)$  is such that

$$\int_0^1 g(t) dt = \int_0^1 f(t) dt$$

and that  $g^{(i)}(0) = g^{(i)}(1) = 0$  if  $0 \leq i \leq L$ . Integrating by parts, we can check that the Fourier coefficients are decreasing as  $C/k^L$ . If  $f$  is singular at the boundaries, the same kind of decay can be achieved but with a slower speed [7]. The algorithm can now be used on the periodised function  $g$ . Due to the special properties of trigonometric functions we have proved in [14], that  $K(p) = O(p)$ , that  $C(p) \leq 1$  and that  $\mu(p)$  does not depend on  $p$ . This enables the convergence of the algorithm with only a small number of random drawings at each step. We have shown furthermore that the rate of convergence of the method for numerical integration is  $1/N^{L-1/2-\epsilon}$ ,  $\forall \epsilon > 0$ . The only drawback is the bad effect of periodisation on the constant  $C$ .

### 3.3. Legendre polynomial basis

The Legendre polynomials  $L_n$  are the orthogonal polynomials on  $[-1, 1]$  with respect to the inner product  $\langle P, Q \rangle = \int_{-1}^1 P(x)Q(x) dx$  such that  $L_0(x) = 1$  and  $L_n(1) = 1$ . Their norms are given by

$$\left( \int_{-1}^1 L_n^2(x) dx \right)^{1/2} = \left( \frac{1}{n + \frac{1}{2}} \right)^{1/2}$$

and they verify the differential equation

$$\frac{d}{dx} ((1 - x^2)L'_n) + n(n + 1)L_n = 0,$$

which is the main tool used to study the quality of the approximation on the Legendre polynomial basis. If we assume that  $f$  belongs to  $C^{2m}([-1, 1])$ , the coefficients  $a_n$  of the mean square approximations on the normalised Legendre polynomials  $\tilde{L}_n$  verify  $|a_n| \leq C_1/n^{2m}$  where  $C_1$  is a constant depending only on  $f$  [3]. The same kind of result will be of course valid on  $[0, 1]$ . The polynomial decay of the coefficients of the approximation is hence achieved without the need of periodisation. On the other hand, the situation concerning the constants  $\tau(p)$ ,  $\mu(p)$  but mainly  $C(p)$  and  $K(p)$  is not as favourable as with the use of the Fourier basis.  $K(p) = O(p^2)$  and  $C(p)$  grows very quickly when  $p$  increases [14]. The number of sample values used at each step of the algorithm has to be greater to ensure its convergence but for a given value of  $p$ , the approximation of a regular function and of its integral are a lot more accurate.

### 3.4. Tchebychef polynomial basis

Tchebychef polynomials are the orthogonal polynomials on  $[-1, 1]$  with respect to the inner product  $\langle P, Q \rangle = \int_{-1}^1 P(x)Q(x)/(\sqrt{1 - x^2}) dx$ . Their expressions are given by  $T_n(x) = \cos(n \arccos(x))$ . It shows that  $\|T_0\|_{L^2_\omega}^2 = \pi$  and  $\|T_n\|_{L^2_\omega}^2 = \pi/2$  if  $n \geq 1$ . They also verify the following differential equation:

$$\frac{d}{dx} (\sqrt{1 - x^2}T'_n(x)) + n^2 \frac{T_n(x)}{\sqrt{1 - x^2}} = 0.$$

If we assume that  $f$  belongs to  $C^{2m}([-1, 1])$ , the coefficients  $a_n$  of the mean square approximations on the normalised Tchebychef polynomials  $\tilde{T}_n$  verify  $|a_n| \leq C_1/n^{2m}$  where  $C_1$  is a constant depending only on  $f$ . This shows that they have the same properties as the Legendre polynomials and

because of their trigonometric expressions, we can expect that the convergence of the algorithm is as fast as with the Fourier basis. It is actually the case, if we compute

$$\langle f, \tilde{T}_n \rangle = \int_{-1}^1 \frac{f(x)\tilde{T}_n(x)}{\sqrt{1-x^2}} dx$$

using the Monte Carlo approximation of  $E(\pi f(V)\tilde{T}_n(V))$  where the density of  $V$  is  $(1/(\pi\sqrt{1-v^2}))1_{[-1,1]}(v)$ . One can check using the same kind of computation as with the Fourier basis that  $K(p) = O(p)$ , that  $C(p)$  is equal to 1 and that  $\mu(p)$  is independent of  $p$  [14]. This version of the algorithm reconciles the good properties of the two others.

#### 4. Approximation on multivariate Fourier basis

##### 4.1. Periodisation in multidimensional integration

We want to compute

$$\int_D f(x_1, x_2, \dots, x_Q) dx_1 dx_2 \dots dx_Q$$

for a function  $f \in C^\infty(D)$ . Even though  $f$  is very smooth, it has to be periodised to be well approximated by its Fourier expansion. Using the same polynomial  $P$  as in dimension one, we transform the previous integral into

$$\int_D f(P(t_1), P(t_2), \dots, P(t_Q))P'(t_1)P'(t_2) \dots P'(t_Q) dt_1 dt_2 \dots dt_Q.$$

Integrating by parts, we can now easily check that the complex Fourier coefficients  $a_m$  of the function

$$g(t_1, \dots, t_Q) = f(P(t_1), P(t_2), \dots, P(t_Q))P'(t_1)P'(t_2) \dots P'(t_Q)$$

verify  $\forall m \in \mathbb{Z}^Q$

$$|a_m| \leq \frac{\beta}{(\tilde{m}_1 \dots \tilde{m}_Q)^L}$$

with  $\tilde{m} = \sup(1, |m|)$  and where  $\beta$  is a positive constant. We can now use our algorithm to approximate the function  $g$ .

##### 4.2. Choice of the basis functions

The approximation of  $g$  on the complex Fourier basis is

$$g(t) = \sum_{m \in \mathbb{Z}^Q} a_m e^{2i\pi m \cdot t}.$$

We can easily see that the most significant coefficients  $a_m$  are the ones for which the product  $(\tilde{m}_1 \dots \tilde{m}_Q)$  is as small as possible. To make the selection of the coefficients to keep according to

this criterion, we define the following sets:

$$Z_{Q,d} = \{m \in \mathbb{Z}^Q / (\widetilde{m}_1 \dots \widetilde{m}_Q) = d\}, \quad V_{Q,d} = \{m \in \mathbb{Z}^Q / (\widetilde{m}_1 \dots \widetilde{m}_Q) \leq d\}.$$

The sets  $Z_{Q,d}$  are the Zaremba crosses which are useful to build lattice rules for numerical integration. The error estimation of these lattice rules can be found in [20] and further developments in [9]. We will give a similar error estimation for both numerical integration and approximation using only uniform random drawings on  $D$ . We now write  $g$  as

$$g(t) = \sum_{m \in V_{Q,d}} a_m e^{2i\pi m \cdot t} + r(t)$$

with

$$r(t) = \sum_{m \in V_{Q,d}^c} a_m e^{2i\pi m \cdot t}.$$

### 4.3. Estimation of the truncation error

This estimation relies mainly on the control of  $\int_D |r(t)|^2 dt$  which can be done using the following lemmas.

**Lemma 4.1.** *We have  $\text{Card}(Z_{Q,j}) \leq 3^Q d(j)^Q$  where  $d(j)$  is the number of divisors of  $j$ .*

**Proof.** We define the sets

$$S_{Q,j} = \{m \in \mathbb{N}_*^Q / m_1 m_2 \dots m_Q = j\}.$$

If  $m_k = 1$ , there are 3 values in  $\mathbb{Z}$ ,  $-1, 0$  and  $1$  such that  $\widetilde{m}_k = 1$ . If  $m_k \neq 1$ , there are 2 values in  $\mathbb{Z}$ ,  $-m_k$  and  $m_k$  such that  $\widetilde{m}_k = m_k$ . As  $m$  has  $Q$  components, we have

$$\text{Card}(Z_{Q,j}) \leq 3^Q \text{Card}(S_{Q,j}).$$

If we now take a component  $m_k$  of  $m$  belonging to  $S_{Q,j}$ , we have at most  $d(j)$  choices for this component and at last

$$\text{Card}(Z_{Q,j}) \leq 3^Q d(j)^Q. \quad \square$$

**Lemma 4.2.**  $\forall \varepsilon > 0 \exists C_2(\varepsilon)$  such that  $d(j) \leq C_2(\varepsilon)j^\varepsilon$ .

**Proof.** See [6].  $\square$

**Lemma 4.3.** *Under the previous assumptions,  $\forall \varepsilon > 0$ , there is a constant  $C_{Q,\varepsilon}$  depending only on  $Q$  and  $\varepsilon$  such that*

$$\int_D |r(t)|^2 dt \leq \frac{C_{Q,\varepsilon}}{d^{2L-1-\varepsilon}}.$$

**Proof.**

$$\int_D |r(t)|^2 dt = \sum_{m \in V_d^c} a_m^2 \leq \beta^2 \sum_{j=d+1}^{\infty} \sum_{m \in Z_{Q,j}} \frac{1}{(\widetilde{m}_1 \dots \widetilde{m}_Q)^{2L}} \leq \beta^2 \sum_{j=d+1}^{\infty} \frac{\text{Card}(Z_{Q,j})}{j^{2L}}$$

which gives from Lemma 4.1

$$\int_D |r(t)|^2 dt \leq \beta^2 \sum_{j=d+1}^{\infty} \frac{3^Q d(j)^Q}{j^{2L}}$$

From Lemma 4.2,  $\forall \varepsilon > 0$ , there is a constant  $\theta(\varepsilon)$  such that

$$d(j) \leq \theta(\varepsilon) j^{\varepsilon/Q}$$

and hence

$$\int_D |r(t)|^2 dt \leq 3^Q \beta^2 \theta(\varepsilon)^Q \sum_{j=d+1}^{\infty} \frac{1}{j^{2L-\varepsilon}}$$

which implies the existence of a constant  $\sigma_{Q,\varepsilon}$  verifying

$$\int_D |r(t)|^2 dt \leq \frac{\sigma_{Q,\varepsilon}}{d^{2L-1-\varepsilon}}. \quad \square$$

#### 4.4. Convergence of the algorithm

We now give the performances of the approximation

$$g^{(M)}(t) = \sum_{m \in V_{Q,d}} a_m^{(M)} e_m(t)$$

keeping the same notations as in Theorem 3.1 and computing the  $P_{Q,d} = \text{Card}(V_{Q,d})$  coefficients which belong to  $V_{Q,d}$ . Instead of using complex Fourier coefficients, we will use real ones which are products of sine–cosine functions in assessing that in the elements  $e_m(t)$  of the approximation basis, sine functions will correspond to negative values of the  $m_i$  and cosine functions to positive ones.

**Theorem 4.4.** *Assuming the previous hypotheses and that*

(i)

$$\frac{P_{Q,d}}{N} < 1,$$

(ii)

$$\tau = \sup \left( \sup_k \int_D (1 - e_k^2(x))^2 dx, \sup_{j,k,j \neq k} \int_D e_k^2(x) e_j^2(x) dx \right) \leq \frac{N}{4},$$

we have

$$E(a_k^{(M)}) = a_k, \quad \text{Var}(a_k^{(M)}) \leq 2 \left( \mu_1 \frac{K(P_{Q,d})^{M-1}}{N^M} C(P_{Q,d})^M + \mu(P_{Q,d}) \frac{1}{d^{2L-1-\varepsilon}} \right)$$

and also

$$E \left( \int_D (g(x) - g^{(M)}(x))^2 dx \right) \leq P_{Q,d} \text{Var}(a_k^{(M)}) + \frac{\mu_2}{d^{2L-1-\varepsilon}}$$

where  $\mu(P_{Q,d})$ ,  $\mu_1$ ,  $\mu_2$ ,  $K(P_{Q,d})$  and  $C(P_{Q,d})$  are positive constants.

**Proof.** From Lemma 2.2, we have

$$\text{Var}(T_m^{(2)}) \leq 2 \left( \frac{\gamma(P_{Q,d})}{N} + \frac{P_{Q,d}^2}{N^2} \gamma_1(P_{Q,d})n \right) \int_D r^2(x) dx$$

which gives from Lemma 4.3

$$\text{Var}(T_m^{(2)}) \leq 2 \left( \frac{\gamma(P_{Q,d})}{N} + \frac{P_{Q,d}^2}{N^2} \gamma_1(P_{Q,d}) \right) \sigma_{Q,\varepsilon} \frac{1}{d^{2L-1-\varepsilon}}$$

and at last as  $P_{Q,d}/N < 1$ ,

$$\text{Var}(T_m^{(2)}) \leq \mu(P_{Q,d}) \frac{1}{d^{2L-1-\varepsilon}},$$

where  $\mu(P_{Q,d}) = 2\sigma_{Q,\varepsilon}(\gamma(P_{Q,d}) + \gamma_1(P_{Q,d}))$  is a positive constant. This inequality is true for two steps. We can easily check by induction that it is still valid if we use  $M$  steps of the algorithm which shows that

$$\text{Var}(T_m^{(M)}) \leq \mu(P_{Q,d}) \frac{1}{d^{2L-1-\varepsilon}}.$$

We can now study  $\text{Var}(a_m)$  at the step  $M$ . We have

$$\text{Var}(a_m^M) = E \left( \left( \sum_{j \in V_{Q,d}} Q_{m,j}^{(M)} a_m - a_m + T_m^{(M)} \right)^2 \right)$$

and hence

$$\text{Var}(a_m^M) \leq 2E \left( \left( \sum_{j \in V_{Q,d}} Q_{m,j}^{(M)} a_m - a_m \right)^2 \right) + E((T_m^{(M)})^2).$$

For the sake of simplicity, we will now write instead  $Q_{m,m}^{(M)}$  of  $Q_{m,m}^{(M)} - 1$ . We have

$$E \left( \left( \sum_{j \in V_{Q,d}} Q_{m,j}^{(M)} a_m \right)^2 \right) \leq \beta^2 E \left( \left( \sum_{j \in V_{Q,d}} |Q_{m,j}^{(M)}| \frac{1}{(\tilde{j}_1 \dots \tilde{j}_Q)^L} \right)^2 \right)$$

that is from the Cauchy–Schwarz inequality and from Lemma 2.1

$$E \left( \left( \sum_{j \in V_{Q,d}} Q_{m,j}^{(M)} a_m \right)^2 \right) \leq \beta^2 \frac{K(P_{Q,d})^{M-1}}{N^M} C(P_{Q,d})^M \left( \sum_{j \in V_{Q,d}} \frac{1}{(\tilde{j}_1 \dots \tilde{j}_Q)^L} \right)^2$$

and as  $L > 1$

$$E \left( \left( \sum_{j \in V_{Q,d}} Q_{m,j}^{(M)} a_m \right)^2 \right) \leq \mu_1 \frac{K(P_{Q,d})^{M-1}}{N^M} C(P_{Q,d})^M.$$

We finally have

$$\text{Var}(a_m^{(M)}) \leq 2 \left( \mu_1 \frac{K(P_{Q,d})^{M-1}}{N^M} C(P_{Q,d})^M + \mu(P_{Q,d}) \frac{1}{d^{2L-1-\varepsilon}} \right).$$

We can now look at the mean square error between  $g$  and its approximation  $g^{(M)}$  computed by the preceding method which is given by

$$E((g(x) - g^{(M)}(x))^2) = E \left( \sum_{m \in V_{Q,d}} (a_m - a_m^{(M)})^2 \right) + \int_D r^2(x) dx,$$

that is,

$$E \left( \int_D (g(x) - g^{(M)}(x))^2 dx \right) \leq \sum_{m \in V_{Q,d}} \text{Var}(a_m^{(M)}) + \int_D r^2(x) dx.$$

As a conclusion, we have

$$E \left( \int_D (g(x) - g^{(M)}(x))^2 dx \right) \leq P_{Q,d} \text{Var}(a_k^{(M)}) + \frac{\mu_2}{d^{2L-1-\varepsilon}}. \quad \square$$

**Remark 4.5.** The error estimations are quite similar to the one we achieved in dimension one. There are nevertheless two drawbacks linked to the dimensional effect. The first one is of course the increase of the number  $P_{Q,d}$  of coefficients that have to be computed to achieve a good approximation. The second one is the very quick increase of the constant  $\beta$  when the degree of periodisation  $L$  and the dimension  $Q$  increase. This will be confirmed in the numerical experiments.

#### 4.5. Numerical results

The algorithm gives us an approximation of the periodised transform  $g$  of the function  $f$ . The integral of  $f$  can be computed either as a particular coefficient of the approximation  $g^M$  of  $g$  or by using the control variate method. The complexity of the algorithm depends mainly on  $P_{Q,d}$ . The following table gives some values of this constant in moderate dimensions.

$d$	$P_{2,d}$	$P_{3,d}$	$P_{4,d}$	$P_{5,d}$
1	9	27	81	243
2	21	81	297	1053
5	61	279	1161	4563
10	149	809	3849	16893
15	241	1391	6945	31743

To see to which dimension  $Q$  we can use the algorithm, we have to compute  $P_{Q,d}$  at least for  $d = 1$  and 2. We obtain

$$P_{Q,1} = 3^Q, \quad P_{Q,2} = P_{Q,1} + 2Q3^{Q-1}.$$

This shows that we cannot take  $Q$  more than about 10, if we do not want to compute more than one million coefficients. We now give numerical results in dimension 3 and 4 to check the algorithm's performance. We will try using these results to find an efficient way to adjust all the parameters which intervene in the algorithm like the number of steps, the degree of periodisation and the number of sample values at each step.

##### 4.5.1. Influence of periodisation

We first test the influence of periodisation. The degree of the polynomial which is useful for periodisation cannot be as high as we wish. Indeed, the constant  $\beta$  grows quickly when this degree increases. This is true in dimension one [7] and even more in higher dimensions so that usually  $L$  is taken equal to 3 or 4 [4]. We will check this result by computing  $\sigma^2 = \int_D (g(x) - g^{(M)}(x))^2 dx$  for different values of  $L$  in taking sufficiently many steps and sample values by step so that only the truncation error remains. This will be done for the function  $f(x, y, z) = \exp(x)\exp(y)\exp(z)$  in the following table:

$d, L$	2	3	4	5	7
1	$3.5 \cdot 10^0$	$7.1 \cdot 10^0$	$2.3 \cdot 10^1$	$5.4 \cdot 10^1$	$1.6 \cdot 10^2$
2	$1.2 \cdot 10^{-1}$	$3.0 \cdot 10^{-1}$	$1.5 \cdot 10^0$	$4.8 \cdot 10^0$	$2.3 \cdot 10^1$
5	$3.3 \cdot 10^{-3}$	$7.7 \cdot 10^{-3}$	$4.6 \cdot 10^{-2}$	$2.1 \cdot 10^{-1}$	$2.2 \cdot 10^0$
10	$8.8 \cdot 10^{-5}$	$6.7 \cdot 10^{-5}$	$4.0 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$	$6.6 \cdot 10^{-2}$
15	$1.5 \cdot 10^{-5}$	$7.1 \cdot 10^{-6}$	$1.7 \cdot 10^{-5}$	$1.3 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$

Numerical results shows that for small values of  $d$ ,  $L = 2$  is preferable. When  $d$  is a little bigger  $L = 3$  seems to be a better choice. This corresponds to a use of a polynomial of degree 7 for periodisation. The value  $L = 3$  will be taken in the following except for singular integrals for which higher degree of periodisation is required [7].

#### 4.5.2. Automatic control of the number of steps

The number of sample values at each step has to be large enough so that the variance is diminishing. This can be ensured in taking for example  $N = 2P_{Q,d}$ . To control the number of steps, we will use a stopping test based on the variance. Every 5 steps, we test if the variance has decreased enough, if not the algorithm stops. We obtain the following results on the previous example.

$d$	1	2	5	10	15
$M$	10	15	25	30	35
$\sigma^2$	$6.1 \cdot 10^0$	$3.5 \cdot 10^{-2}$	$7.2 \cdot 10^{-3}$	$5.9 \cdot 10^{-5}$	$7.3 \cdot 10^{-6}$

These results shows that the number of steps has been optimised for a given accuracy. We will use from now on this version of the algorithm.

#### 4.5.3. Numerical performances of the algorithm

To check the performances of the algorithm, we will compute the variance and two approximate values of the integral for different values of  $d$ . The first one will be the coefficient  $e_0(g)$  and the second one  $\tilde{I}$  will be achieved using the control variate method with 10 000 sample values. The CPU times will be given in seconds in all the numerical examples.

**Example 4.6.** We begin with the function  $f(x, y, z) = 1/8\sqrt{xyz}$  which has singularities in each variable at 0, which has been periodised with a polynomial of degree 15 and whose integral is equal to 1. The variance is infinite before periodisation and worth about 6 after.

$d$	$P_{3,d}$	$ I - e_0(g) $	$\sigma^2$	$ I - \tilde{I} $	CPU
1	27	$1.8 \cdot 10^{-2}$	$1.2 \cdot 10^0$	$4.3 \cdot 10^{-3}$	$2.0 \cdot 10^{-1}$
2	81	$7.6 \cdot 10^{-3}$	$2.7 \cdot 10^{-2}$	$6.8 \cdot 10^{-4}$	$8.0 \cdot 10^{-1}$
5	279	$8.4 \cdot 10^{-4}$	$7.0 \cdot 10^{-4}$	$8.5 \cdot 10^{-5}$	$6.4 \cdot 10^0$
10	809	$1.1 \cdot 10^{-4}$	$8.6 \cdot 10^{-6}$	$2.6 \cdot 10^{-5}$	$4.9 \cdot 10^1$
15	1391	$1.5 \cdot 10^{-5}$	$8.3 \cdot 10^{-7}$	$6.5 \cdot 10^{-6}$	$1.4 \cdot 10^2$

Numerical results are similar with  $f(x, y, z) = \exp(x)\exp(y)\exp(z)$  which confirms that the singularities have been eliminated. CPU time is acceptable.

**Example 4.7.** We now take from [4] the example in dimension 4 of the function  $f(x, y, z, t) = \exp(xyzt)$  whose integral is about 1.04. Its variance before periodisation is about  $1.3 \times 10^{-2}$  and

about 7 after. We have the following table:

$d$	$P_{4,d}$	$ I - e_0(g) $	$\sigma^2$	$ I - \tilde{I} $	CPU
1	81	$6.0 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$3.8 \cdot 10^{-3}$	$1.2 \cdot 10^0$
2	297	$1.4 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	$9.1 \cdot 10^0$
5	1161	$2.9 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	$9.7 \cdot 10^{-5}$	$1.3 \cdot 10^2$
10	3849	$4.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$3.9 \cdot 10^{-5}$	$1.3 \cdot 10^3$

CPU times are getting too large when  $d$  increases. Periodisation has increased artificially the variance and the number of terms in the approximation is too large compared to the variance reduction. Hence, we will now try to achieve a better approximation using multivariate Legendre polynomials for which no periodisation is required.

## 5. Approximation on multivariate Legendre polynomials

### 5.1. Decay of the coefficients

We will assume that  $f$  belongs to  $C^{2k}([-1, 1]^Q)$ . Its approximation is given by

$$f(x) = \sum_{m \in \mathbb{N}^Q} a_m e_m(x),$$

where

$$e_m(x) = \widetilde{L}_{m_1}(x_1) \widetilde{L}_{m_2}(x_2) \dots \widetilde{L}_{m_Q}(x_Q).$$

We achieve the same kind of decay for the coefficients  $a_m$  as with the Fourier basis. We have

$$|a_m| \leq \frac{C_1}{(\widehat{m}_1 \widehat{m}_2 \dots \widehat{m}_Q)^{2k}},$$

where  $C_1$  is a positive constant and with  $\widehat{m} = \sup(1, m)$ . This is achieved by using the same reasoning as in dimension one for each of the  $Q$  integration variables. The main tool is still the differential equation satisfied by the  $L_n$ . The same decay is obviously also valid on  $D = [0, 1]^Q$  and we will now assume that

$$|a_m| \leq \frac{C_1}{(\widehat{m}_1 \widehat{m}_2 \dots \widehat{m}_Q)^L}.$$

### 5.2. Choice of the elements of the approximation basis

We can easily check that the most significant coefficients are the ones for which the product

$$\widehat{m}_1 \widehat{m}_2 \dots \widehat{m}_Q$$

is as small as possible. To search the coefficients to keep according to this criterion, we now define, as we have done for the Fourier basis, the following sets:

$$T_{Q,d} = \{m \in \mathbb{N}^Q / (\widehat{m}_1 \dots \widehat{m}_Q) = d\}, \quad W_{Q,d} = \{m \in \mathbb{N}^Q / (\widehat{m}_1 \dots \widehat{m}_Q) \leq d\}.$$

We can now give an error estimation similar to the one achieved with the Fourier basis. We have

$$f(t) = \sum_{m \in W_{Q,d}} a_m e_m(t) + \sum_{m \in W_{Q,d}^C} a_m e_m(t)$$

and we will now put

$$r(t) = \sum_{m \in W_{Q,d}^C} a_m e_m(t).$$

### 5.3. Estimation of the truncation error

We give the control on  $\int_D r(t)^2 dt$  using the same tools as in Lemma 4.3.

**Lemma 5.1.** *Under the previous assumptions,  $\forall \varepsilon > 0$ , there is a constant  $C_{Q,\varepsilon}$  depending only on  $Q$  and  $\varepsilon$  such that*

$$\int_D r(t)^2 dt \leq \frac{C_{Q,\varepsilon}}{d^{2L-1-\varepsilon}}.$$

**Proof.** One only has to say that

$$W_{Q,d} \subset V_{Q,d}. \quad \square$$

### 5.4. Convergence of the algorithm

We now give the performances of the approximation

$$f^{(M)}(t) = \sum_{m \in W_{Q,d}} a_m^{(M)} e_m(t)$$

keeping the same notations as in Theorem 3.1 and computing the  $L_{Q,d} = \text{Card}(W_{Q,d})$  coefficients which belong to  $W_{Q,d}$ .

**Theorem 5.2.** *Assuming the previous hypotheses and that*

(i)

$$\frac{L_{Q,d}}{N} < 1,$$

(ii)

$$\tau = \sup \left( \sup_k \int_D (1 - e_k^2(x))^2 dx, \sup_{j,k,j \neq k} \int_D e_k^2(x)e_j^2(x) dx \right) \leq \frac{N}{4},$$

we have

$$E(a_k^{(M)}) = a_k, \quad \text{Var}(a_k^{(M)}) \leq 2 \left( \mu_1 \frac{K(L_{Q,d})^{M-1}}{N^M} C(L_{Q,d})^M + \mu(L_{Q,d}) \frac{1}{d^{2L-1-\varepsilon}} \right)$$

and also

$$E \left( \int_D (f(x) - f^{(M)}(x))^2 dx \right) \leq L_{Q,d} \text{Var}(a_k^{(M)}) + \frac{\mu_2}{d^{2L-1-\varepsilon}},$$

where  $\mu(L_{Q,d}), \mu_1, \mu_2, K(L_{Q,d})$  and  $C(L_{Q,d})$  are positive constants.

**Proof.** See Theorem 4.4.  $\square$

### 5.5. Numerical results

The algorithm gives us an approximation of the function  $f$ . The integral of  $f$  can be computed either as a peculiar coefficient of the approximation  $g^M$  of  $g$  or by using the control variate method. The complexity of the algorithm depends mainly on  $L_{Q,d}$ . The following table gives some values of this constant in moderate dimensions.

$d$	$L_{2,d}$	$L_{3,d}$	$L_{4,d}$	$L_{5,d}$	$L_{6,d}$
1	4	8	16	32	64
2	8	20	48	112	256
5	21	62	168	432	1072
10	48	165	504	1432	3872
15	76	276	880	2592	7232

To see to which dimension  $Q$  we can use the algorithm, we have to compute  $L_{Q,d}$  at least for  $d = 1$  and 2. We obtain

$$L_{Q,1} = 2^Q, \quad L_{Q,2} = L_{Q,1} + Q2^Q.$$

This shows that we cannot take  $Q$  more than about 15, if we do not want to compute more than one million coefficients. There are less coefficients to compute than with the Fourier basis and the approximation should also be more accurate if we refer to what happened in dimension one. There is nevertheless one main drawback for this new approach. The number of sample values at each step has to be larger to ensure the convergence of the algorithm. We will set this parameter to  $50L_{Q,d}$ . This will be enough to overcome the problems linked to the quick increase of  $C(L_{Q,d})$  with  $Q$  and

$d$  and that  $K(L_{Q,d})$  is a  $O(L_{Q,d}^2)$  on all the numerical examples that will be studied. To check the performances of the algorithm, we will compute the variance and two approximate values of the integral for different values of  $d$ . The first one will be the coefficient  $e_0(f)$  and the second one  $\tilde{I}$  will be achieved using the control variate method with 10 000 sample values.

**Example 5.3.** We first study the function  $f(x, y, z) = \exp(x)\exp(y)\exp(z)$  whose integral is about 5 and the variance 7.

$d$	$L_{3,d}$	$ I - e_0(f) $	$\sigma^2$	$ I - \tilde{I} $	CPU
1	8	$2.2 \cdot 10^{-1}$	$1.8 \cdot 10^0$	$1.6 \cdot 10^{-3}$	$1.0 \cdot 10^{-1}$
2	20	$4.7 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	$2.7 \cdot 10^{-4}$	$2.0 \cdot 10^{-1}$
5	62	$2.1 \cdot 10^{-5}$	$2.1 \cdot 10^{-6}$	$2.3 \cdot 10^{-5}$	$1.4 \cdot 10^0$
10	165	$4.4 \cdot 10^{-7}$	$1.3 \cdot 10^{-9}$	$1.4 \cdot 10^{-7}$	$1.9 \cdot 10^1$
15	276	$1.2 \cdot 10^{-8}$	$1.4 \cdot 10^{-11}$	$4.6 \cdot 10^{-8}$	$7.5 \cdot 10^1$

We will use these results to make the comparison between our algorithm and crude Monte Carlo integration. CPU time to make  $10^8$  random drawings using this method is about 30 s. The relative accuracy is  $10^{-4}\sigma_f$ . This corresponds to the accuracy achieved with our algorithm if we take  $d = 2$  for a CPU time of only 0.2 s. Comparisons are even more favourable to our algorithm when  $d$  increases.

**Example 5.4.** We now take the example of the function  $f(x, y, z, t) = \exp(xyzt)$  which has already been studied using the Fourier basis.

$d$	$L_{4,d}$	$ I - e_0(f) $	$\sigma^2$	$ I - \tilde{I} $	CPU
1	16	$1.2 \cdot 10^{-4}$	$8.0 \cdot 10^{-5}$	$9.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-1}$
2	48	$2.0 \cdot 10^{-5}$	$4.9 \cdot 10^{-6}$	$6.7 \cdot 10^{-5}$	$1.0 \cdot 10^0$
5	168	$3.2 \cdot 10^{-6}$	$2.5 \cdot 10^{-7}$	$2.1 \cdot 10^{-5}$	$1.1 \cdot 10^1$
10	504	$4.2 \cdot 10^{-7}$	$1.1 \cdot 10^{-8}$	$2.1 \cdot 10^{-6}$	$1.6 \cdot 10^2$

For a given  $d$ , we achieve a much better accuracy than with the Fourier basis. The number of basis functions to achieve the approximation is also lower. This is crucial when we use the control variate method once the approximation has been computed. The gain in terms of variance reduction needs to compensate the additional cost due to the pointwise computation of the approximation function  $\tilde{f}$ . Even if we do not take into account the additional time to compute this approximation, we must have anyway

$$\frac{\text{Time}(\tilde{f})}{\text{Time}(f)} \leq \frac{\sigma_f^2}{\sigma_{\tilde{f}}^2},$$

where  $\text{Time}(f)$  is the pointwise CPU time of the function  $f$ . If we take  $d = 5$ , we have  $\sigma_f^2/\sigma_{\tilde{f}}^2 \simeq 10\,000$  and  $\text{Time}(\tilde{f})/\text{Time}(f) \simeq 200$  which shows that this criterion is respected here.

**Example 5.5.** We take at last from [11] the example of the function  $f(x, y, z, t, u, v) = \exp((x + y + z + t + u + v)/6)$  whose integral is around 1.66 and the variance  $3.9 \times 10^{-2}$ .

$d$	$L_{6,d}$	$ I - e_0(f) $	$\sigma^2$	$ I - \tilde{I} $	CPU
1	64	$4.5 \cdot 10^{-5}$	$1.9 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$	$1.6 \cdot 10^0$
2	256	$2.4 \cdot 10^{-7}$	$3.7 \cdot 10^{-9}$	$6.0 \cdot 10^{-7}$	$3.4 \cdot 10^1$
3	448	$2.2 \cdot 10^{-8}$	$5.3 \cdot 10^{-11}$	$2.7 \cdot 10^{-8}$	$1.4 \cdot 10^2$
5	1072	$4.2 \cdot 10^{-10}$	$3.3 \cdot 10^{-14}$	$3.0 \cdot 10^{-10}$	$1.4 \cdot 10^3$

Numerical results are still satisfying and even more accurate than in the previous example. We can nevertheless notice that the dimensional effect is now getting really disturbing. Only small values of  $d$  should be used and the value  $d = 5$  should not exceed especially in even higher dimensions. We can also notice that the introduction of the sets  $W_{Q,d}$  reduces drastically the number of terms in the approximation basis compared to a simple Cartesian product. For example, the number of terms in the approximation is 1072 for  $d = 5$  instead of  $46566 = 6^6$ . The use of Legendre polynomials has enabled to get rid of the problems linked to the periodisation method occurring with the Fourier basis but has increased the constants  $K(p)$  and  $C(p)$ .

## 6. Approximation on multivariate Tchebychef polynomials

We now present the version of the algorithm based on multivariate Tchebychef polynomials. As in dimension one, we can expect that this version reconciles the good properties of the two previous versions. This will be confirmed in the numerical results. We will also compare our algorithm to the method developed in [1] which reaches an optimal rate of convergence for regular functions.

### 6.1. Numerical results

The approximation algorithm has to be slightly modified in order to compute the coefficients of the approximation. One has to use importance sampling according to the weight function  $\omega(x) = 1/(\sqrt{1-x^2})$  for each of the integration variables. The choice of the elements in the multivariate Tchebychef polynomial approximation is effected using the same criteria as with the Legendre polynomials. The decay of the coefficients is achieved using the differential equation satisfied by the  $T_n$ . We will now use the algorithm with only  $6L_{Q,d}$  random drawings instead of  $50L_{Q,d}$  with the Legendre polynomials. We will first make a comparison with some of the previous examples after having linearly transformed the integrals from  $[0, 1]^Q$  into  $[-1, 1]^Q$ . The approximation  $f^{(M)}$  of  $f$  can be written

$$f^{(M)}(x) = \sum_{k \in W_{Q,d}} a_k^{(M)} e_k(x),$$

where the  $a_k^{(M)}$  are approximations of the  $\langle f, e_k \rangle$  and the  $e_k$  the elements of the approximation basis. It is very satisfying, but it does not directly give the value of

$$I(f) = \int_D f(x) dx.$$

There are two ways to obtain this value. We can either compute

$$\tilde{I}(f) = \sum_{k \in W_{Q,d}} a_k^{(M)} \int_D e_k(x) dx,$$

or we can use the control variate method with  $f^{(M)}$  as an approximation of  $f$ . This will lead to an approximate integral  $\bar{I}(f)$  using 10 000 sample values and to an approximate value of the variance  $\sigma^2$ .

**Example 6.1.** We begin with  $f(x, y, z) = \exp(x) \exp(y) \exp(z)$  in the following table:

$d$	$L_{3,d}$	$ I - \tilde{I} $	$\sigma^2$	$ I - \bar{I} $	CPU
5	62	$9.8 \cdot 10^{-5}$	$6.0 \cdot 10^{-6}$	$8.2 \cdot 10^{-6}$	$4.0 \cdot 10^{-1}$
10	165	$2.5 \cdot 10^{-6}$	$3.6 \cdot 10^{-9}$	$3.3 \cdot 10^{-7}$	$3.0 \cdot 10^0$
15	276	$4.0 \cdot 10^{-8}$	$5.3 \cdot 10^{-11}$	$2.4 \cdot 10^{-8}$	$9.4 \cdot 10^0$
20	411	$6.5 \cdot 10^{-9}$	$2.6 \cdot 10^{-13}$	$1.4 \cdot 10^{-8}$	$2.3 \cdot 10^1$
25	546	$1.8 \cdot 10^{-9}$	$5.9 \cdot 10^{-14}$	$3.0 \cdot 10^{-9}$	$4.1 \cdot 10^1$

For a given value of  $d$ , we achieve almost the same accuracy as with Legendre polynomials. The slight difference may be explained because the integral is not directly given by one of the coefficients of the approximation. On the other hand, CPU is roughly 8 times smaller which corresponds to the ratio of the sample values used for each of the two algorithms. We can increase  $d$  to achieve even more accuracy with still lower CPU times.

**Example 6.2.** We take once more the example of the function  $f(x, y, z, t, u, v) = \exp((x + y + z + t + u + v)/6)$  in the following table:

$d$	$L_{6,d}$	$ I - \tilde{I} $	$\sigma^2$	$ I - \bar{I} $	CPU
2	256	$6.0 \cdot 10^{-6}$	$4.8 \cdot 10^{-9}$	$1.7 \cdot 10^{-6}$	$1.8 \cdot 10^1$
3	448	$8.2 \cdot 10^{-7}$	$2.6 \cdot 10^{-10}$	$8.0 \cdot 10^{-8}$	$5.4 \cdot 10^1$
5	1072	$6.0 \cdot 10^{-9}$	$6.5 \cdot 10^{-14}$	$2.0 \cdot 10^{-9}$	$3.7 \cdot 10^2$

The conclusions are the same as with the previous example. We achieve slightly less accurate results with significantly lower CPU times.

### 6.2. Comparison with an optimal method

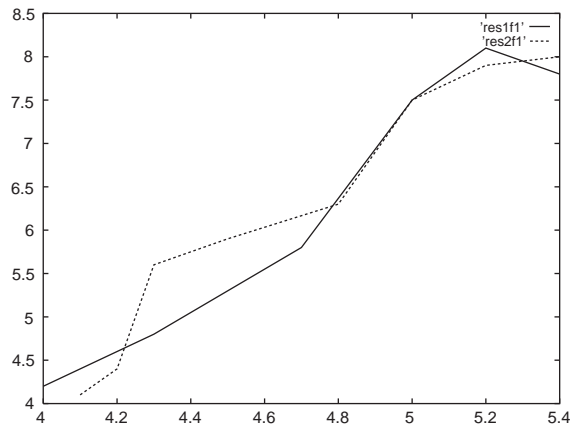
We will now compare our method to the one developed in [1] whose rate of convergence is

$$\frac{1}{N^{1/2+k/Q}}$$

for  $C^k$  functions in dimension  $Q$ . The principle of this method is to divide the integration domain into  $n^Q$  cubes, to compute the interpolation polynomial at  $C_Q^{k+Q-1}$  well chosen deterministic points and then use the control variate method with a small number of points  $m$  on each of the cubes. This

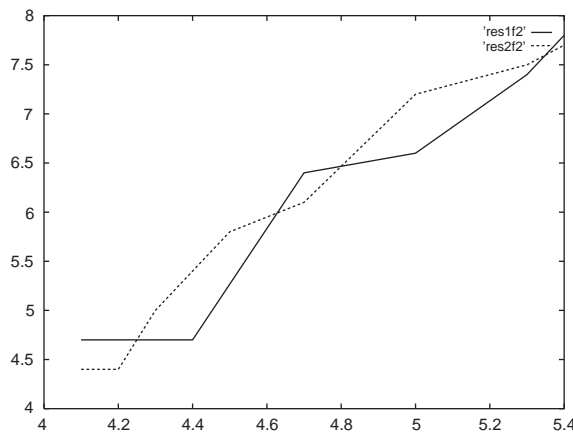
corresponds in dimension 4 to about  $N = 4 \times 10^5$  integration points if  $n = 10$  and to 2 millions points if  $n = 15$ . The relative accuracies are respectively  $10^{-7}$  and  $10^{-8}$  on the following examples. We will plot  $-\log_{10} |I - \tilde{I}|/|I|$  and  $-\log_{10} |I - \tilde{I}|/|I|$  as a function of  $\log_{10}(N_t)$  where  $N_t$  is a number of sample values used in the approximation algorithm. We can compare with the previous results and give an approximate value of the order of the method by computing the slope of the curve.

**Example 6.3.** We begin with the example of the function  $f(x, y, z, t) = \exp(x + y + z + t)$  which leads to the following graphic:



We achieve an accuracy of about 8 digits with about  $4 \times 10^5$  sample values. This is quite similar to the results achieved in [1]. An approximate value of the slope is 1.4 which is a lot better than 0.5 with crude Monte Carlo.

**Example 6.4.** We now take the example of  $f(x, y, z, t) = \exp(x) \sin(y) \cos(z) \log(1 + t)$  which leads to the following curve.



Numerical results are analogous to the previous ones and confirm the efficiency of the algorithm on an example where integration variables do not play the same role.

## 7. Conclusion

We have built and studied an iterative algorithm which appears to be a very efficient tool in the numerical computation of the approximation  $\tilde{f}$  of a regular function  $f$  in moderate dimensions. As an application, the approximate value of  $\int_D f(x) dx$  is given either by  $\int_D \tilde{f}(x) dx$  or by using the control variate method. The accuracy achieved using the version of the algorithm based on Tchebychef polynomials is almost optimal. The next step is to try to optimise the algorithm. This can surely be done by replacing the random drawings by law discrepancy sequences and by making some data storage to transform one step of the algorithm into a cubature formula. We can also add that the approximation algorithm can certainly be used to build Galerkin methods for integral equations, spectral methods, spectral finite elements and numerical schemes for Kinetic equations.

## 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. Beekers, A. Haegmans, Transformations of integrands for lattice rules, in: T.O. Espelid, A. Genz (Eds.), *Numerical Integration-Recent Developments, Software and Applications*, Kluwer, Dordrecht, Netherlands, 1992, pp. 329–340.
- [3] C. Bernardi, Y. Maday, *Approximations spectrales de problèmes aux limites elliptiques*, Springer, Berlin, 1992.
- [4] P. Davis, P. Rabinowitz, *Methods of Numerical Integration*, 2nd Edition, Computer Science and Applied Mathematics, Academic Press, New York, 1984.
- [5] S. Haber, A modified Monte-Carlo quadrature, *Math. Comput.* 20 (1966) 361–368.
- [6] G.H. Hardy, E.M. Wright, *An introduction to the Theory of Numbers*, 5th Edition, Oxford University Press, Oxford, 1979.
- [7] P. Helluy, S. Maire, P. Ravel, Intégration numérique d'ordre élevé de fonctions régulières ou singulières sur un intervalle, *C. R. Acad. Sci. Paris, Sér. I* 327 (1998) 843–848, *Analyse Numérique*.
- [8] M. Iri, S. Moriguti, Y. Takazawa, On a Certain Quadrature Formula, Vol. 91, *Kokyuroku of Research, Institute for Mathematic Science, Kyoto University*, 1970, pp. 82–118 (in Japanese).
- [9] S. Joe, I. Sloan, Imbedded lattice rules for multidimensionnal integration, *SIAM J. Numer. Anal.* 29 (4) (1992) 1119–1135.
- [10] M.H. Kalos, P.A. Whitlock, *Monte Carlo Methods*, Wiley, New York, 1986.
- [11] A.R. Krommer, C.W. Ueberhuber, *Computational Integration*, SIAM, Philadelphia, 1998.
- [12] B. Lapeyre, E. Pardoux, R. Sentis, *Méthodes de Monte-Carlo pour les équations de transport et de diffusion*, Springer, Berlin, 1998.
- [13] G.P. Lepage, A new algorithm for adaptative multidimensional integration, *J. Comput. Phys.* 27 (1978) 192–203.
- [14] S. Maire, Reducing variance using iterated control variates, *J. Statistic. Comput. Simulation* 73 (2003) 1–29.
- [15] H. Niederreiter, Quasi-Monte Carlo methods and pseudorandom numbers, *Bull. Amer. Math. Soc.* 84 (1978) 957–1041.
- [16] A. Philippe, Processing simulation output by Riemann sums, *J. Statistic. Comput. Simulation* 59 (1997) 295–314.
- [17] W.H. Press, G.R. Farrar, Recursive stratified sampling for multidimensional Monte Carlo integration, *Comput. Phys.* 4 (1990) 190–195.
- [18] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in FORTRAN*, Cambridge University Press, Cambridge, 1996.
- [19] T.W. Sag, G. Szekeres, Numerical evaluation of high-dimensional integrals, *Math. Comput.* 18 (1964) 245–253.
- [20] I.H. Sloan, P.J. Kachoyan, Lattice methods for multiple integration: theory, error analysis and examples, *SIAM J. Numer. Anal.* 24 (1987) 116–128.