Multidimensional Sensitivity Analysis of Large-Scale Mathematical Models

Ivan Dimov and Rayna Georgieva

Abstract Sensitivity analysis (SA) is a procedure for studying how sensitive are the output results of large-scale mathematical models to some uncertainties of the input data. The models are described as a system of partial differential equations. Often such systems contain a large number of input parameters. Obviously, it is important to know how sensitive is the solution to some uncontrolled variations or uncertainties in the input parameters of the model. Algorithms based on analysis of variances technique for calculating numerical indicators of sensitivity and computationally efficient Monte Carlo integration techniques have recently been developed by the authors. They have been successfully applied to sensitivity studies of air pollution levels calculated by the Unified Danish Eulerian Model with respect to several important input parameters. In this paper a comprehensive theoretical and experimental study of the Monte Carlo algorithm based on *symmetrised shaking* of Sobol sequences has been done. It has been proven that this algorithm has an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error. Extensive numerical experiments with Monte Carlo, quasi-Monte Carlo (QMC) and scrambled QMC algorithms based on Sobol sequences are performed to support the theoretical studies and to analyze applicability of the algorithms to various classes of problems. The numerical tests show that the Monte Carlo algorithm based on *symmetrised shaking* of Sobol sequences gives reliable results for multidimensional integration problems under consideration.

Keywords Global sensitivity analysis • Analysis for independent inputs • Monte Carlo and quasi-Monte Carlo algorithms • Optimal Monte Carlo methods

I. Dimov (✉) • R. Georgieva
Department of Parallel Algorithms, IICT, Bulgarian Academy
of Sciences, Acad. G. Bonchev 25 A, 1113 Sofia, Bulgaria
e-mail: ivdimov@bas.bg; rayna@parallel.bas.bg

Mathematics Subject Classification (2010): 49Q12, 65C05, 65B99

1 Introduction

Most existing methods for providing SA rely on special assumptions connected to the behavior of the model (such as linearity, monotonicity and additivity of the relationship between model input and model output) [22]. Such assumptions are often applicable to a large range of mathematical models. At the same time there are models that include significant nonlinearities and/or stiffness. For such models assumptions about linearity and additivity are not applicable. This is especially true when one deals with nonlinear systems of partial differential equations. The numerical study and results reported in this paper have been done by using a large-scale mathematical model called Unified Danish Eulerian Model (UNI-DEM) [33, 34]. The model enables us to study the transport of air pollutants and other species over a large geographical region. The system of partial differential equations describes the main physical processes, such as advection, diffusion, deposition as well as chemical and photochemical processes between the studied species. The emissions and the quickly changing meteorological conditions are also described. The nonlinearity of the equations is mainly introduced when modeling chemical reactions [33]. If the model results are sensitive to a given process, one can describe it mathematically in a more adequate way or more precisely. Thus, the goal of our study is to increase the reliability of the results produced by the model and to identify processes that must be studied more carefully, as well as to find input parameters that need to be measured with a higher precision. A careful sensitivity analysis is needed in order to decide where and how simplifications of the model can be made. That is why it is important to develop and study more adequate and reliable methods for sensitivity analysis. A good candidate for reliable sensitivity analysis of models containing nonlinearity is the variance-based method [22]. The idea of this approach is to estimate how the variation of an input parameter or a group of inputs contributes into the variance of the model output. As a measure of this analysis we use the *total sensitivity indices (TSI)* (see, Sect. 2) described as multidimensional integrals:

$$I = \int_{\Omega} g(\mathrm{x})p(\mathrm{x})\,\mathrm{d}\mathrm{x}, \ \Omega \subset \mathbf{R}^d, \tag{1}$$

where $g(\mathrm{x})$ is a square integrable function in Ω and $p(\mathrm{x}) \geq 0$ is a *probability density function* (p.d.f.), such that $\int_{\Omega} p(\mathrm{x})\,\mathrm{d}\mathrm{x} = 1$.

Clearly, the progress in the area of sensitivity analysis is closely related to the progress in reliable algorithms for multidimensional integration.

2 Problem Setting

2.1 Modeling and Sensitivity

Assume that the mathematical model can be presented as a function

$$u = f(x), \quad \text{where} \quad x = (x_1, x_2, \ldots, x_d) \in U^d \equiv [0; 1]^d \tag{2}$$

is the vector of input parameters with a joint p.d.f. $p(x) = p(x_1, \ldots, x_d)$. Assume also that the input variables are independent (noncorrelated) and the density function $p(x)$ is known, even if x_i are not actually random variables (r.v.). The TSI of an input parameter $x_i, i \in \{1, \ldots, d\}$ is defined in the following way [9, 26]:

$$S_i^{tot} = S_i + \sum_{l_1 \neq i} S_{il_1} + \sum_{l_1, l_2 \neq i, l_1 < l_2} S_{il_1 l_2} + \ldots + S_{il_1 \ldots l_{d-1}}, \tag{3}$$

where S_i is called *the main effect (first-order sensitivity index)* of x_i and $S_{il_1 \ldots l_{j-1}}$ is the *j*-th order sensitivity index. The higher-order terms describe the interaction effects between the unknown input parameters $x_{i_1}, \ldots, x_{i_v}, v \in \{2, \ldots, d\}$ on the output variance.

The method of global SA used in this work is based on a decomposition of an integrable model function f in the d-dimensional factor space into terms of increasing dimensionality [26]:

$$f(x) = f_0 + \sum_{v=1}^{d} \sum_{l_1 < \ldots < l_v} f_{l_1 \ldots l_v}(x_{l_1}, x_{l_2}, \ldots, x_{l_v}), \tag{4}$$

where f_0 is a constant. The representation (4) is referred to as the ANOVA representation of the model function $f(x)$ if each term is chosen to satisfy the following condition [26]:

$$\int_0^1 f_{l_1 \ldots l_v}(x_{l_1}, x_{l_2}, \ldots, x_{l_v}) dx_{l_k} = 0, \quad 1 \leq k \leq v, \quad v = 1, \ldots, d.$$

Let us mention the fact that if the whole presentation (4) of the right-hand side is used, this does not make the problem simpler. The hope is that a truncated sequence $f_0 + \sum_{v=1}^{d_{tr}} \sum_{l_1 < \ldots < l_v} f_{l_1 \ldots l_v}(x_{l_1}, x_{l_2}, \ldots, x_{l_v})$, where $d_{tr} < d$ (or even $d_{tr} \ll d$), can be considered as a good approximation to the model function f.

The quantities

$$\mathbf{D} = \int_{U^d} f^2(x) dx - f_0^2, \quad \mathbf{D}_{l_1 \ldots l_v} = \int f_{l_1 \ldots l_v}^2 dx_{l_1} \ldots dx_{l_v} \tag{5}$$

are the so-called total and partial variances, respectively, and are obtained after squaring and integrating over U^d the equality (4) on the assumption that $f(\mathrm{x})$ is a square integrable function (thus, all terms in (4) are also square integrable functions). Therefore, the total variance of the model output is split into partial variances in the analogous way as the model function, that is, the unique ANOVA-decomposition: $\mathbf{D} = \sum_{\nu=1}^{d} \sum_{l_1 < \ldots < l_\nu} \mathbf{D}_{l_1 \ldots l_\nu}$. The use of probability theory concepts is based on the assumption that the input parameters are random variables distributed in U^d that defines $f_{l_1 \ldots l_\nu}(x_{l_1}, x_{l_2}, \ldots, x_{l_\nu})$ also as random variables with variances (5). For example, f_{l_1} is presented by a conditional expectation: $f_{l_1}(x_{l_1}) = \mathbf{E}(\mathrm{u}|x_{l_1}) - f_0$ and, respectively, $\mathbf{D}_{l_1} = \mathbf{D}[f_{l_1}(x_{l_1})] = \mathbf{D}[\mathbf{E}(\mathrm{u}|x_{l_1})]$. Based on these assumptions about the model function and the output variance, the following quantities

$$S_{l_1 \ldots l_\nu} = \frac{\mathbf{D}_{l_1 \ldots l_\nu}}{\mathbf{D}}, \quad \nu \in \{1, \ldots, d\} \tag{6}$$

are referred to as the global sensitivity indices [26]. Based on the formulas (5)–(6), it is clear that the mathematical treatment of the problem of providing global sensitivity analysis consists in evaluating total sensitivity indices (3) of corresponding order that, in turn, leads to computing multidimensional integrals of the form (1). It means that to obtain S_i^{tot} in general, one needs to compute 2^d (or $2^{d_{tr}}$, with $d_{tr} \ll d$) integrals of type (5).

The procedure for computing global sensitivity indices (see [26]) is based on the following representation of the variance:

$$\mathbf{D}_\mathrm{y} : \mathbf{D}_\mathrm{y} = \int f(\mathrm{x}) \, f(\mathrm{y}, \mathrm{z}') \mathrm{dxdz}' - f_0^2, \tag{7}$$

where $\mathrm{y} = (x_{k_1}, \ldots, x_{k_m})$, $1 \le k_1 < \ldots < k_m \le d$, is an arbitrary set of m variables $(1 \le m \le d-1)$ and z is the set of $d-m$ complementary variables, i.e. $\mathrm{x} = (\mathrm{y}, \mathrm{z})$. The equality (7) enables the construction of a Monte Carlo algorithm for evaluating $f_0, \mathbf{D}$ and $\mathbf{D}_\mathrm{y}$:

$$\frac{1}{n} \sum_{j=1}^{n} f(\xi_j) \xrightarrow{P} f_0, \qquad \frac{1}{n} \sum_{j=1}^{n} f(\xi_j) \, f(\eta_j, \zeta_j') \xrightarrow{P} \mathbf{D}_\mathrm{y} + f_0^2,$$

$$\frac{1}{n} \sum_{j=1}^{n} f^2(\xi_j) \xrightarrow{P} \mathbf{D} + f_0^2, \qquad \frac{1}{n} \sum_{j=1}^{n} f(\xi_j) \, f(\eta_j', \zeta_j) \xrightarrow{P} \mathbf{D}_\mathrm{z} + f_0^2,$$

where $\xi = (\eta, \zeta)$ is a random sample and η corresponds to the input subset denoted by y.

Instead of randomized (Monte Carlo) algorithms for computing the above sensitivity parameters, one can use deterministic quasi-Monte Carlo (QMC) algorithms or randomized QMC [13, 14]. Randomized (Monte Carlo) algorithms have proven to be very efficient in solving multidimensional integrals in composite domains [3, 23]. At the same time the QMC based on well-distributed Sobol sequences

can be considered as a good alternative to Monte Carlo algorithms, especially for smooth integrands and not very high *effective dimensions* (up to $d = 15$) [12]. Sobol $\Lambda\Pi_\tau$ are good candidates for efficient QMC algorithms. Algorithms based on $\Lambda\Pi_\tau$ sequences while being deterministic mimic the pseudorandom sequences used in Monte Carlo integration. One of the problems with $\Lambda\Pi_\tau$ sequences is that they may have *bad* two-dimensional projection. In this context *bad* means that the distribution of the points is far from being a uniform distribution. If such projections are used in a certain computational problem, then the lack of uniformity may provoke a substantial lost of accuracy. To overcome this problem randomized QMC can be used. There are several ways of randomization and *scrambling* is one of them. The original motivation of scrambling [10, 19] aims toward obtaining more uniformity for quasi-random sequences in high dimensions, which can be checked via two-dimensional projections. Another way of randomisation is to *shake* the quasi-random points according to some procedure. Actually, the scrambled algorithms obtained by *shaking* the quasi-random points can be considered as Monte Carlo algorithms with a special choice of the density function. It is a matter of definition. Thus, there is a reason to be able to compare two classes of algorithms: *deterministic* and *randomized*.

3 Complexity in Classes of Algorithms

One may pose the task to consider and compare two classes of algorithms: *deterministic algorithms* and *randomized (Monte Carlo) algorithms*. Let I be the desired value of the integral. Assume for a given r.v. θ one can prove that the mathematical expectation satisfies $\mathbf{E}\theta = I$. Suppose that the mean value of n values of θ: $\theta^{(i)}$, $i = 1, \ldots, n$ is considered as a Monte Carlo approximation to the solution: $\bar{\theta}_n = 1/n \sum_{i=1}^{n} \theta^{(i)} \approx I$, where $\theta^{(i)} (i = 1, 2, \ldots, n)$ correspond to values (realizations) of a r.v. θ. In general, a certain randomized algorithm can produce the result with a given probability error. So, dealing with randomized algorithms one has to accept that the result of the computation can be true only with a certain (although high) probability. In most practical computations it is reasonable to accept an error estimate with a probability smaller than 1.

Consider the following integration problem:

$$S(f) := I = \int_{U^d} f(\mathbf{x}) d\mathbf{x}, \tag{8}$$

where $\mathbf{x} \equiv (x_1, \ldots, x_d) \in U^d \subset \mathbf{R}^d$ and $f \in C(U^d)$ is an integrable function on U^d. The computational problem can be considered as a mapping of function f : $\{[0,1]^d \to \mathbf{R}\}$ to $\mathbf{R}$: $S(f) : f \to \mathbf{R}$, where $S(f) = \int_{U^d} f(\mathbf{x}) d\mathbf{x}$ and $f \in F_0 \subset C(U^d)$. We refer to S as the solution operator. The elements of F_0 are the data, for which the problem has to be solved, and for $f \in F_0$, $S(f)$ is the exact solution. For a given f, we want to compute exactly or approximately $S(f)$. One may be interested in cases

when the integrand f has a higher regularity. It is because in many cases of practical computations f is smooth and has high-order bounded derivatives. If this is the case, then is it reasonable to try to exploit such a smoothness. To be able to do that we need to define the functional class $F_0 \equiv \mathbf{W}^k(\|f\|; U^d)$ in the following way:

Definition 3.1. Let d and k be integers, $d, k \geq 1$. We consider the class $\mathbf{W}^k(\|f\|; U^d)$ (sometimes abbreviated to $\mathbf{W}^k$) of real functions f defined over the unit cube $U^d = [0,1)^d$, possessing all the partial derivatives $\dfrac{\partial^r f(\mathbf{x})}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}$, $\alpha_1 + \cdots + \alpha_d = r \leq k$, which are continuous when $r < k$ and bounded in sup norm when $r = k$. The seminorm $\|\cdot\|$ on $\mathbf{W}^k$ is defined as

$$\|f\| = \sup \left\{ \left| \frac{\partial^k f(\mathbf{x})}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}} \right|, \ \alpha_1 + \cdots + \alpha_d = k, \ \mathbf{x} \equiv (x_1, \dots, x_d) \in U^d \right\}.$$

We keep the seminorm $\|f\|$ into the notation for the functional class $\mathbf{W}^k(\|f\|; U^d)$ since it is important for our further consideration. We call a *quadrature formula* any expression of the form

$$A^D(f, n) = \sum_{i=1}^{n} c_i f(\mathbf{x}^{(i)}),$$

which approximates the value of the integral $S(f)$. The real numbers $c_i \in \mathbf{R}$ are called weights and the d-dimensional points $\mathbf{x}^{(i)} \in U^d$ are called nodes. It is clear that for fixed weights c_i and nodes $\mathbf{x}^{(i)} \equiv (x_{i,1}, \dots, x_{i,d})$, the quadrature formula $A^D(f, n)$ may be used to define an algorithm with an integration error $err(f, A^D) \equiv \int_{U^d} f(\mathbf{x}) d\mathbf{x} - A^D(f, n)$. We call a *randomized quadrature formula* any formula of the following kind: $A^R(f, n) = \sum_{i=1}^{n} \sigma_i f(\xi^{(i)})$, where σ_i and $\xi^{(i)}$ are random weights and nodes, respectively. The algorithm $A^R(f, n)$ belongs to the class of randomized (Monte Carlo) denoted by $\mathscr{A}^{\mathscr{R}}$.

Definition 3.2. Given a randomized (Monte Carlo) integration formula for the functions from the space $\mathbf{W}^k$, we define the integration error

$$err(f, A^R) \equiv \int_{U^d} f(\mathbf{x}) d\mathbf{x} - A^R(f, n)$$

by the probability error $\varepsilon_P(f)$ in the sense that $\varepsilon_P(f)$ is the least possible real number, such that

$$Pr\left(\left| err(f, A^R) \right| < \varepsilon_P(f) \right) \geq P,$$

and the mean square error

$$r(f) = \left\{ E\left[err^2(f, A^R) \right] \right\}^{1/2}.$$

We assume that it suffices to obtain an $\varepsilon_P(f)$-approximation to the solution with a probability $0 < P < 1$. If we allow equality, i.e. $0 < P \leq 1$ in Definition 3.2, then

$\varepsilon_P(f)$ can be used as an accuracy measure for both randomized and deterministic algorithms. In such a way it is consistent to consider a wider class $\mathscr{A}$ of algorithms that contains both classes: randomized and deterministic algorithms.

Definition 3.3. Consider the set $\mathscr{A}$ of algorithms A:

$$\mathscr{A} = \{A : Pr(|err(f,A)| \leq \varepsilon) \geq c\}, \quad A \in \{A^D, A^R\}, \ 0 < c < 1$$

that solve a given problem with an integration error $err(f,A)$.

In such a setting it is correct to compare randomized algorithms with algorithms based on low-discrepancy sequences like Sobol $\Lambda\Pi_\tau$ sequences.

4 The Algorithms

The algorithms we study are based on Sobol $\Lambda\Pi_\tau$ sequences.

4.1 $\Lambda\Pi_\tau$ Sobol Sequences

$\Lambda\Pi_\tau$ sequences are *uniformly distributed sequences* (u.d.s.) The term *u.d.s.* was introduced by Hermann Weyl in 1916 [30]. For practical purposes a u.d.s. should satisfy the following three requirements [23, 25]: (i) the best asymptote as $n \to \infty$, (ii) well-distributed points for small n and (iii) a computationally inexpensive algorithm.

All $\Lambda\Pi_\tau$ sequences given in [25] satisfy the first requirement. Suitable distributions such as $\Lambda\Pi_\tau$ sequences are also called (t,m,s)-nets and (t,s)-sequences in base $b \geq 2$. To introduce them, define first an elementary s-interval in base b as a subset of U^s of the form $E = \prod_{j=1}^s \left[\frac{a_j}{b^{d_j}}, \frac{a_j+1}{b^{d_j}} \right]$, where $a_j, d_j \geq 0$ are integers and $a_j < b^{d_j}$ for all $j \in \{1,\ldots,s\}$. Given two integers $0 \leq t \leq m$, a (t,m,s)-net in base b is a sequence $\mathrm{x}^{(i)}$ of b^m points of U^s such that $Card\ E \cap \{\mathrm{x}^{(1)},\ldots,\mathrm{x}^{(b^m)}\} = b^t$ for any elementary interval E in base b of hypervolume $\lambda(E) = b^{t-m}$. Given a non-negative integer t, a (t,s)-sequence in base b is an infinite sequence of points $\mathrm{x}^{(i)}$ such that for all integers $k \geq 0, m \geq t$, the sequence $\{\mathrm{x}^{(kb^m)},\ldots,\mathrm{x}^{((k+1)b^m-1)}\}$ is a (t,m,s)-net in base b.

Sobol [23] defines his Π_τ-meshes and $\Lambda\Pi_\tau$ sequences, which are (t,m,s)-nets and (t,s)-sequences in base 2, respectively. The terms (t,m,s)-nets and (t,s)-sequences in base b (also called Niederreiter sequences) were introduced in 1988 by Niederreiter [18].

To generate the j-th component of the points in a Sobol sequence, we need to choose a primitive polynomial of some degree s_j over the Galois field of two elements GF(2) $P_j = x^{s_j} + a_{1,j}x^{s_j-1} + a_{2,j}x^{s_j-2} + \ldots + a_{s_j-1,j}x + 1$, where

the coefficients $a_{1,j}, \ldots, a_{s_j-1,j}$ are either 0 or 1. A sequence of positive integers $\{m_{1,j}, m_{2,j}, \ldots\}$ are defined by the recurrence relation

$$m_{k,j} = 2a_{1,j}m_{k-1,j} \oplus 2^2 a_{2,j}m_{k-2,j} \oplus \ldots \oplus 2^{s_j}m_{k-s_j,j} \oplus m_{k-s_j,j},$$

where $\oplus$ is the bit-by-bit *exclusive-or* operator. The values $m_{1,j}, \ldots, m_{s_j,j}$ can be chosen freely provided that each $m_{k,j}, 1 \le k \le s_j$, is odd and less than 2^k. Therefore, it is possible to construct different Sobol sequences for the fixed dimension s. In practice, these numbers must be chosen very carefully to obtain really efficient Sobol sequence generators [27]. The so-called direction numbers $\{v_{1,j}, v_{2,j}, \ldots\}$ are defined by $v_{k,j} = \dfrac{m_{k,j}}{2^k}$. Then the j-th component of the i-th point in a Sobol sequence is given by $x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \ldots$, where i_k is the k-th binary digit of $i = (\ldots i_3 i_2 i_1)_2$. Subroutines to compute these points can be found in [2, 24]. The work [15] contains more details.

4.2 The Monte Carlo Algorithms Based on Modified Sobol Sequences: MCA-MSS

One of the algorithms based on a procedure of *shaking* was proposed recently in [6]. The idea is that we take a Sobol $\Lambda\Pi_\tau$ point (vector) x of dimension d. Then x is considered as a centrum of a sphere with a radius $\rho < 1$. A random point $\xi \in U^d$ uniformly distributed on the sphere is taken. Consider a random variable θ defined as a value of the integrand at that random point, i.e. $\theta = f(\xi)$. Consider random points $\xi^{(i)}(\rho) \in U^d, i = 1, \ldots, n$. Assume $\xi^{(i)}(\rho) = x^{(i)} + \rho\omega^{(i)}$, where $\omega^{(i)}$ is a unique uniformly distributed vector in U^d. The radius ρ is relatively small $\rho \ll \frac{1}{2^{d_j}}$, such that $\xi^{(i)}(\rho)$ is still in the same elementary i-th interval $E_i^d = \prod_{j=1}^d \left[\frac{a_j^{(i)}}{2^{d_j}}, \frac{a_j^{(i)}+1}{2^{d_j}} \right]$, where the pattern $\Lambda\Pi_\tau$ point $x^{(i)}$ is. We use a subscript i in E_i^d to indicate that the i-th $\Lambda\Pi_\tau$ point $x^{(i)}$ is in it. So, we assume that if $x^{(i)} \in E_i^d$, then $\xi^{(i)}(\rho) \in E_i^d$ too.

It was proven in [6] that the mathematical expectation of the random variable $\theta = f(\xi)$ is equal to the value of the integral (8), that is, $\mathbf{E}\theta = S(f) = \int_{U^d} f(x)dx$. This result allows for defining a randomized algorithm. One can take the Sobol $\Lambda\Pi_\tau$ point $x^{(i)}$ and *shake* it somewhat. *Shaking* means to define random points $\xi^{(i)}(\rho) = x^{(i)} + \rho\omega^{(i)}$ according to the procedure described above. For simplicity the algorithm described above is abbreviated as MCA-MSS-1.

The probability error of the algorithm MCA-MSS-1 was analysed in [7]. It was proved that for integrands with continuous and bounded first derivatives, i.e. $f \in \mathbf{W}^1(L; U^d)$, where $L = \|f\|$, it holds

$$err(f,d) \le c_d' \|f\| n^{-\frac{1}{2}-\frac{1}{d}} \quad \text{and} \quad r(f,d) \le c_d'' \|f\| n^{-\frac{1}{2}-\frac{1}{d}},$$

where the constants c_d' and c_d'' do not depend on n.

In this work a modification of algorithm MCA-MSS-1 is proposed and analysed. The new algorithm will be called MCA-MSS-2.

It is assumed that $n = m^d$, $m \geq 1$. The unit cube U^d is divided into m^d disjoint subdomains, such that they coincide with the elementary d-dimensional subintervals defined in Sect. 4.1 $U^d = \bigcup_{j=1}^{m^d} K_j$, where $K_j = \prod_{i=1}^{d} [a_i^{(j)}, b_i^{(j)})$, with $b_i^{(j)} - a_i^{(j)} = \dfrac{1}{m}$ for all $i = 1, \ldots, d$.

In such a way in each d-dimensional subdomain K_j, there is exactly one $\Lambda \Pi_\tau$ point $\mathrm{x}^{(j)}$. Assuming that after shaking, the random point stays inside K_j, i.e. $\xi^{(j)}(\rho) = \mathrm{x}^{(j)} + \rho \omega^{(j)} \in K_j$, one may try to exploit the smoothness of the integrand in case if the integrand f belongs to $\mathbf{W}^2(L; U^d)$.

Then, if $p(x)$ is a p.d.f., such that $\int_{U^d} p(\mathrm{x}) d\mathrm{x} = 1$, then

$$\int_{K_j} p(\mathrm{x}) d\mathrm{x} = p_j \leq \frac{c_1^{(j)}}{n},$$

where $c_1^{(j)}$ are constants. If d_j is the diameter of K_j, then

$$d_j = \sup_{x_1, x_2 \in K_j} |x_1 - x_2| \leq \frac{c_2^{(j)}}{n^{1/d}},$$

where $c_2^{(j)}$ are another constants.

In the particular case when the subintervals are with edge $1/m$ for all constants, we have $c_1^{(j)} = 1$ and $c_2^{(j)} = \sqrt{d}$. In each subdomain K_j the central point is denoted by $s^{(j)}$, where $s^{(j)} = (s_1^{(j)}, s_2^{(j)}, \ldots, s_d^{(j)})$.

Suppose two random points $\xi^{(j)}$ and $\xi^{(j)'}$ are chosen, such that $\xi^{(j)}$ is selected during our procedure used in MCA-MSS-1. The second point $\xi^{(j)'}$ is chosen to be symmetric to $\xi^{(j)}$ according to the central point $s^{(j)}$ in each cube K_j. In such a way the number of random points is $2m^d$. One may calculate all function values $f(\xi^{(j)})$ and $f(\xi^{(j)'})$, for $j = 1, \ldots, m^d$, and approximate the value of the integral in the following way:

$$I(f) \approx \frac{1}{2m^d} \sum_{j=1}^{2n} \left[f(\xi^{(j)}) + f(\xi^{(j)'}) \right]. \tag{9}$$

This estimate corresponds to MCA-MSS-2. We prove later on that this algorithm has an optimal rate of convergence for functions with bounded second derivatives, i.e. for functions $f \in \mathbf{W}^2(L; U^d)$, while the algorithm MCA-MSS-1 has an optimal rate of convergence for functions with bounded first derivatives: $f \in \mathbf{W}^1(L; U^d)$.

One can prove the following:

Theorem 1. *The quadrature formula (9) constructed above for integrands f from* $\mathbf{W}^2(L;U^d)$ *satisfies*

$$err(f,d) \leq \tilde{c}_d' \|f\| n^{-\frac{1}{2}-\frac{2}{d}}$$

and

$$r(f,d) \leq \tilde{c}_d'' \|f\| n^{-\frac{1}{2}-\frac{2}{d}},$$

where the constants $\tilde{c}_d'$ *and* $\tilde{c}_d''$ *do not depend on n.*

Proof. One can see that

$$\mathbf{E}\left\{\frac{1}{2m^d}\sum_{j=1}^{2n}\left[f(\xi^{(j)})+f(\xi^{(j)'})\right]\right\} = \int_{U^d} f(\mathbf{x})d\mathbf{x}.$$

For the fixed $\Lambda\Pi_\tau$ point $\mathbf{x}^{(j)} \in K_j$ one can use the d-dimensional Taylor formula to present the function $f(\mathbf{x}^{(j)})$ in K_j around the central point $\mathbf{s}^{(j)}$. Since $f \in \mathbf{W}^2(L;U^d)$, there exists a d-dimensional point $\eta^{(j)} \in K_j$ lying between $\mathbf{x}^{(j)}$ and $\mathbf{s}^{(j)}$ such that

$$f(\mathbf{x}^{(j)}) = f(\mathbf{s}^{(j)}) + \nabla f(\mathbf{s}^{(j)})\,(\mathbf{x}^{(j)}-\mathbf{s}^{(j)})$$
$$+ \frac{1}{2}(\mathbf{x}^{(j)}-\mathbf{s}^{(j)})^T\,[D^2 f(\eta^{(j)})](\mathbf{x}^{(j)}-\mathbf{s}^{(j)}), \quad (10)$$

where $\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1},\ldots,\frac{\partial f(\mathbf{x})}{\partial x_d}\right]$ and $[D^2 f(\mathbf{x})] = \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_k}\right]_{i,k=1}^d$. For simplicity the superscript of the argument (j) in the last two formulas is omitted assuming that the formulas are written for the j-th cube K_j. Now, we can write formula (10) at previously defined random points ξ and ξ' both belonging to K_j. In such a way we have

$$f(\xi) = f(\mathbf{s}) + \nabla f(\mathbf{s})\,(\xi-\mathbf{s}) + \frac{1}{2!}(\xi-\mathbf{s})^T[D^2 f(\eta)](\xi-\mathbf{s}), \quad (11)$$

$$f(\xi') = f(\mathbf{s}) + \nabla f(\mathbf{s})\,(\xi'-\mathbf{s}) + \frac{1}{2!}(\xi'-\mathbf{s})^T[D^2 f(\eta')](\xi'-\mathbf{s}), \quad (12)$$

where η' is another d-dimensional point lying between ξ' and $\mathbf{s}$. Adding (11) and (12), we get

$$f(\xi)+f(\xi') = 2f(\mathbf{s}) + \frac{1}{2}\left\{(\xi-\mathbf{s})^T[D^2 f(\eta)](\xi-\mathbf{s}) + \right.$$
$$\left. + (\xi'-\mathbf{s})^T[D^2 f(\eta')]\,(\xi'-\mathbf{s})\right\}.$$

Because of the symmetry there is no term depending on the gradient $Df(s)$ in the previous formula. If we consider the variance $\mathbf{D}[f(\xi) + f(\xi')]$ taking into account that the variance of the constant $2f(s)$ is zero, then we get

$$\mathbf{D}[f(\xi) + f(\xi')] =$$

$$= \mathbf{D}\left\{ \frac{1}{2} \left[(\xi - s)^T [D^2 f(\eta)](\xi - s) + (\xi' - s)^T [D^2 f(\eta')](\xi' - s) \right] \right\}$$

$$\leq \mathbf{E}\left\{ \frac{1}{2} \left[(\xi - s)^T [D^2 f(\eta)](\xi - s) + (\xi' - s)^T [D^2 f(\eta')](\xi' - s) \right] \right\}^2.$$

Since $f \in \mathbf{W}^2(L; U^d)$, we can strengthen the last inequality if the terms $[D^2 f(\eta)]$ and $[D^2 f(\eta')]$ are substituted by the seminorm L (and removing front bracket) and the products $(\xi - s)^T (\xi - s)$ and $(\xi' - s)^T (\xi' - s)$ by the squared diameter of the subdomain K_j. Now we return back to the notation with superscript, taking into account that the above consideration is just for an arbitrary subdomain K_j. The variance can be estimated from above in the following way:

$$\mathbf{D}[f(\xi) + f(\xi')] \leq L^2 \sup_{x_1^{(j)}, x_2^{(j)}} \left| x_1^{(j)} - x_2^{(j)} \right|^4 \leq L^2 (c_2^{(j)})^4 n^{-4/d}.$$

Now the variance of $\theta_n = \sum_{j=1}^n \theta^{(j)}$ can be estimated:

$$\mathbf{D}\theta_n = \sum_{j=1}^n p_j^2 \mathbf{D}[f(\xi) + f(\xi')] \leq \sum_{j=1}^n (c_1^{(j)})^2 n^{-2} L^2 (c_2^{(j)})^4 n^{-4/d}$$

$$\leq \left(L c_1^{(j)} c_2^{(j)2} \right)^2 n^{-1-4/d}. \tag{13}$$

Therefore, $r(f, d) \leq \tilde{c}_d'' \|f\| n^{-\frac{1}{2} - \frac{2}{d}}$. The application of Tchebycheff's inequality to the variance (13) yields

$$\varepsilon(f, d) \leq \tilde{c}_d' \|f\| n^{-\frac{1}{2} - \frac{2}{d}}$$

for the probable error ε, where $\tilde{c}_d' = \sqrt{2d}$, which concludes the proof.

One can see that the Monte Carlo algorithm MCA-MSS-2 has an optimal rate of convergence for functions with continuous and bounded second derivative [3]. This means that the rate of convergence $(n^{-\frac{1}{2} - \frac{2}{d}})$ cannot be improved for the functional class $\mathbf{W}^2$ in the class of the randomized algorithms $\mathscr{A}^{\mathscr{R}}$.

Note that both MCA-MSS-1 and MCA-MSS-2 have one control parameter, that is, the radius ρ of the *sphere of shaking*. At the same time, to be able to efficiently use this control parameter, one should increase the computational complexity. The problem is that after *shaking* the random point may leave the multidimensional

subdomain. That is why after each such a procedure, one should be checking if the random point is still in the same subdomain. It is clear that the procedure of checking if a random point is inside the given domain is a computationally expensive procedure when one has a large number of points. A small modification of MCA-MSS-2 algorithm allows to overcome this difficulty. If we just generate a random point $\xi^{(j)} \in K_j$ uniformly distributed inside K_j and after that take the symmetric point $\xi^{(j)'}$ according to the central point $s^{(j)}$, then this procedure will simulate the algorithm MCA-MSS-2. Such a completely randomized approach simulates algorithm MCA-MSS-2, but the *shaking* is with different radiuses ρ in each subdomain. We call this algorithm MCA-MSS-2-S, because this approach looks like the stratified symmetrised Monte Carlo. Obviously, MCA-MSS-2-S is less expensive than MCA-MSS-2, but there is not such a control parameter like the radius ρ, which can be considered as a parameter randomly chosen in each subdomain K_j.

It is important to notice that all three algorithms MCA-MSS-1, MCA-MSS-2 and MCA-MSS-2-S have optimal (unimprovable) rate of convergence for the corresponding functional classes, that is, MCA-MSS-1 is optimal in $\mathbf{W}^1(L;U^d)$ and both MCA-MSS-2 and MCA-MSS-2-S are optimal in $\mathbf{W}^2(L;U^d)$.

We also consider the known Owen nested scrambling algorithm [19] for which it is proved that the rate of convergence is $n^{-3/2}(log\ n)^{(d-1)/2}$, which is very good but still not optimal even for integrands in $\mathbf{W}^1(L;U^d)$. One can see that if the *logarithmic function* from the estimate can be omitted, then the rate will become optimal. Let us mention that it is still not proven that the above estimate is exact, that is, we do not know if the *logarithm* can be omitted. It should be mentioned that the proved convergence rate for the Owen nested scrambling algorithm improves significantly the rate for the unscrambled nets, which is $n^{-1}(log\ n)^{d-1}$. That is why it is important to compare numerically our algorithms MCA-MSS with the Owen nested scrambling. The idea of Owen nested scrambling is based on randomization of a single digit at each iteration. Let $x^{(i)} = (x_{i,1}, x_{i,2}, \ldots, x_{i,s})$, $i = 1, \ldots, n$ be quasi-random numbers in $[0,1)^s$, and let $z^{(i)} = (z_{i,1}, z_{i,2}, \ldots, z_{i,s})$ be the scrambled version of the point $x^{(i)}$. Suppose that each $x_{i,j}$ can be represented in base b as $x_{i,j} = (0.x_{i1,j}\ x_{i2,j} \ldots x_{iK,j} \ldots)_b$ with K being the number of digits to be scrambled. Then nested scrambling proposed by Owen [19, 20] can be defined as follows: $z_{i1,j} = \pi_\bullet(x_{i1,j})$, and $z_{il,j} = \pi_{\bullet x_{i1,j} x_{i2,j} \ldots x_{il-1,j}}(x_{il,j})$, with independent permutations $\pi_{\bullet x_{i1,j} x_{i2,j} \ldots x_{il-1,j}}$ for $l \geq 2$. Of course, (t,m,s)-net remains (t,m,s)-net under nested scrambling. However, nested scrambling requires b^{l-1} permutations to scramble the l-th digit. Owen scrambling (nested scrambling), which can be applied to all (t,s)-sequences, is powerful; however, from the implementation point of view, nested scrambling or so-called path-dependent permutations require a considerable amount of bookkeeping and lead to more problematic implementation. There are various versions of scrambling methods based on digital permutation, and the differences among those methods are based on the definitions of the π_l's. These include Owen nested scrambling [19, 20], Tezuka's generalized Faure sequences [29] and Matousek's linear scrambling [17].

5 Case Study: Variance-Based Sensitivity Analysis of the Unified Danish Eulerian Model

The input data for the sensitivity analysis performed in this paper has been obtained during runs of a large-scale mathematical model for remote transport of air pollutants (UNI-DEM, [33]). The model enables us to study concentration variations in time of a high number of air pollutants and other species over a large geographical region (4,800 × 4,800 km), covering the whole of Europe, the Mediterranean and some parts of Asia and Africa. Such studies are important for environmental protection, agriculture and health care. The model presented as a system of partial differential equations describes the main processes in the atmosphere including photochemical processes between the studied species, the emissions and the quickly changing meteorological conditions. Both nonlinearity and stiffness of the equations are mainly introduced when modeling chemical reactions [33]. The chemical scheme used in the model is the well-known condensed CBM-IV (Carbon Bond Mechanism). Thus, the motivation to choose UNI-DEM is that it is one of the models of atmospheric chemistry, where the chemical processes are taken into account in a very accurate way.

This large and complex task is not suitable for direct numerical treatment. For the purpose of numerical solution, it is split into submodels, which represent the main physical and chemical processes. The sequential splitting [16] is used in the production version of the model, although other splitting methods have also been considered and implemented in some experimental versions [4, 5]. Spatial and time discretization makes each of the above submodels a huge computational task, challenging for the most powerful supercomputers available nowadays. That is why parallelization has always been a key point in the computer implementation of DEM since its very early stages.

Our main aim here is to study the sensitivity of the ozone concentration according to the rate variation of some chemical reactions. We consider the chemical rates to be the input parameters and the concentrations of pollutants to be the output parameters.

6 Numerical Results and Discussion

Some numerical experiments are performed to study experimentally various properties of the algorithms. We are interested in both smooth and non-smooth integrands. The reason to consider both cases is that we deal with many different output functions using the UNI-DEM model. Formally the output functions should have enough smoothness, because the solution has bounded second derivatives by definition. Nevertheless, some functions of concentrations that depend on photochemical reactions in the air have computational irregularities. It means that the derivative of

Table 1 Relative error and computational time for numerical integration of a smooth function
($S(f_2) \approx 0.10897$)

	SFMT		Sobol QMCA		Owen scrambling		MCA-MSS-1		
n	Rel. error	Time (s)	Rel. error	Time (s)	Rel. error	Time (s)	$\rho \times 10^3$	Rel. error	Time (s)
10^2	0.0562	0.002	0.0365	< 0.001	0.0280	0.001	3.9	0.0363	0.001
							13	0.0036	0.001
10^3	0.0244	0.004	0.0023	0.001	0.0016	0.001	1.9	0.0038	0.010
							6.4	0.0019	0.010
10^4	0.0097	0.019	0.0009	0.002	0.0003	0.003	0.8	0.0007	0.070
							2.8	0.0006	0.065

the function is very high by modulo and it causes computational difficulties—the
function behaves as a non-smooth function.

The expectations based on theoretical results are that for non-smooth functions
MCA-MSS algorithms based on the *shaking* procedures outperform the QMC even
for relatively low dimensions. It is also interesting to observe how behave the
randomized QMC based on scrambled Sobol sequences.

For our numerical tests we use the following non-smooth integrand:

$$f_1(x_1,x_2,x_3,x_4) = \sum_{i=1}^{4} |(x_i - 0.8)^{-1/3}|, \tag{14}$$

for which even the first derivative does not exist. Such kinds of applications appear
also in some important problems in financial mathematics. The referent value of the
integral $S(f_1)$ is approximately equal to 7.22261. To make a comparison we also
consider an integral with a smooth integrand:

$$f_2(x_1,x_2,x_3,x_4) = x_1 \, x_2^2 \, \mathrm{e}^{x_1 x_2} \sin x_3 \cos x_4. \tag{15}$$

The second integrand (15) is a function $f_2 \in C^\infty(U^d)$ with a referent value of the
integral $S(f_2)$ approximately equal to 0.10897. The integration domain in both cases
is $U^4 = [0,1]^4$.

Some results from the numerical integration tests with a smooth (15) and a non-
smooth (14) integrand are presented in Tables 1 and 2, respectively. As a measure of
the efficiency of the algorithms, both the relative error (defined as the absolute error
divided by the referent value) and computational time are shown. For generating
Sobol quasi-random sequences, the algorithm with Gray code implementation [1]
and sets of direction numbers proposed by Joe and Kuo [11] are used. The MCA-
MSS-1 algorithm [6] involves generating random points uniformly distributed on a
sphere with radius ρ. One of the best available random number generators, SIMD-
oriented Fast Mersenne Twister (SFMT) [21, 32] 128-bit pseudorandom number
generator of period $2^{19937} - 1$ has been used to generate the required random
points. SFMT algorithm is a very efficient implementation of the plain Monte

Table 2 Relative error and computational time for numerical integration of a non-smooth function ($S(f_1) \approx 7.22261$)

n	SFMT Rel. error	SFMT Time (s)	Sobol QMCA Rel. error	Sobol QMCA Time (s)	Owen scrambling Rel. error	Owen scrambling Time (s)	MCA-MSS-1 $\rho \times 10^3$	MCA-MSS-1 Rel. error	MCA-MSS-1 Time (s)
10^3	0.0010	0.011	0.0027	0.001	0.0021	0.002	1.9	0.0024	0.020
							6.4	0.0004	0.025
7.10^3	0.0009	0.072	0.0013	0.009	0.0003	0.011	1.0	0.0004	0.110
							3.4	0.0005	0.114
3.10^4	0.0005	0.304	0.0003	0.032	0.0003	0.041	0.6	0.0001	0.440
							1.9	0.0002	0.480
5.10^4	0.0007	0.513	0.0002	0.053	2e-05	0.066	0.4	7e-05	0.775
							1.4	0.0001	0.788

Carlo method [23]. The radius ρ depends on the integration domain, number of samples and minimal distance between Sobol deterministic points δ. We observed experimentally that the behavior of the relative error of numerical integration is significantly influenced by the fixed radius of spheres. That is why the values of the radius ρ are presented according to the number of samples n used in our experiments, as well as to a fixed coefficient, *radius coefficient* $\kappa = \rho/\delta$. The latter parameter gives the ratio of the radius to the minimal distance between Sobol points. The code of scrambled quasi-random sequences used in our studies is taken from the collection of NAG C Library [31]. This implementation of scrambled quasi-random sequences is based on TOMS Algorithm 823 [10]. In the implementation of the scrambling, there is a possibility to make a choice of three methods of scrambling: the first is a restricted form of Owen scrambling [19], the second is based on the method of Faure and Tezuka [8] and the last method combines the first two (it is referred to as a *combined* approach).

Random points for the MCA-MSS-1 algorithm have been generated using the original Sobol sequences and modeling a random direction in d-dimensional space. The computational time of the calculations with pseudorandom numbers generated by SFMT (see columns labeled as *SFMT* and *MCA-MSS* in Tables 1 and 2) has been estimated for all 10 algorithm runs.

Comparing the results in Tables 1 and 2 one observes that:

- All algorithms under consideration are efficient and converge with the expected rate of convergence.
- In the case of *smooth functions*, the Sobol algorithm is better than SFMT (the relative error is up to 10 times smaller than for SFMT).
- The scrambled QMC and MCA-MSS-1 are much better than the classical Sobol algorithm; in many cases even the simplest *shaking* algorithm MCA-MSS-1 gives a higher accuracy than the scrambled algorithm.
- In the case of *non-smooth functions*, SFMT algorithm implementing the plain Monte Carlo method is better than the Sobol algorithm for relatively small samples (n).

Table 3 Relative error and computational time for numerical integration of a smooth function ($S(f) \approx 0.10897$)

No. of points n (No. of double points $2n$)	Sobol QMCA Rel. error	Time (s)	MCA-MSS-1 $\rho \times 10^3$	Rel. error	Time (s)	MCA-MSS-2 Rel. error	Time (s)	MCA-MSS-2-S Rel. error	Time (s)
2^9	0.0059	< 0.001	2.1	0.0064	0.009	0.0033	0.010	0.0016	0.005
(2×2^9)			6.4	0.0061	0.010	0.0032	0.010		
2^{10}	0.0035	0.002	1.9	0.0037	0.010	9e-05	0.020	0.0002	0.007
(2×2^{10})			6.4	0.0048	0.010	0.0002	0.020		
2^{16}	2e-05	0.027	0.4	3e-05	1.580	7 e-06	1.340	9e-06	0.494
(2×2^{16})			1.2	0.0001	1.630	5e-06	1.380		

- In the case of *non-smooth functions*, our Monte Carlo *shaking* algorithm MCA-MSS-1 gives similar results as the scrambled QMC; for several values of n, we observe advantages for MCA-MSS-1 in terms of accuracy.
- Both MCA-MSS-1 and scrambled QMC are better than SFMT and Sobol quasi MC algorithm in the case of non-smooth functions.

Another observation is that for the chosen integrands the scrambling algorithm does not outperform the algorithm with the original Sobol points, but the scrambled algorithm and Monte Carlo algorithm MCA-MSS-1 are more stable with respect to relative errors for relatively small values of n.

In Table 3 we compare Sobol QMCA with MCA-MSS-2 and MCA-MSS-2-S, as well as with simplest *shaking* algorithm MCA-MSS-1. The results show that the simplest *shaking* algorithm MCA-MSS-1 gives relative errors similar to errors of the Sobol QMCA, which is expected since the $\Lambda \Pi_\tau$ Sobol sequences are already quite well distributed. That is why one should not expect improvement for a very smooth integrand. But the *symmetrised shaking* algorithm MCA-MSS-2 improves the relative error. The effect of this improvement is based on the fact that the second derivatives of the integrand exists, they are bounded and the construction of the MCA-MSS-2 algorithm gives a better convergence rate of order $O(n^{-1/2-2/d})$. The same convergence rate has the algorithm MCA-MSS-2-S, but the latter one does not allow to control the value of the radius of *shaking*. As expected MCA-MSS-2-S gives better results than MCA-MSS-1. The relative error obtained by MCA-MSS-2 and MCA-MSS-2-S are of the same magnitude (see Table 3). The advantage of MCA-MSS-2-S is that its computational complexity is much smaller. A comparison of the relative error and computational complexity for different values of n is presented in Table 4. To have a fair comparison we have to consider again a smooth function (15). The observation is that MCA-MSS-2-S algorithm outperforms the simplest *shaking* algorithm MCA-MSS-1 in terms of relative error and complexity.

After testing the algorithms under consideration on the *smooth* and *non-smooth functions*, we studied the efficiency of the algorithms on *real-life* functions obtained after running UNI-DEM. Polynomials of 4th degree with 35 unknown coefficients are used to approximate the *mesh functions* containing the model outputs.

Table 4 Relative error and computational time for numerical integration of a smooth function ($S(f) \approx 0.10897$) (comparison between MCA-MSS-1 and MCA-MSS-2-S algorithms)

n	Sobol QMCA		MCA-MSS-1			MCA-MSS-2-S	
	Rel. $\times 10^3$	Time error	ρ (s)	Rel. error	Time (s)	Rel. error	Time (s)
2×4^4	0.0076	< 0.001	2.1	0.0079	< 0.001	0.0016	0.005
(512)			6.4	0.0048	< 0.001		
2×6^4	0.0028	0.001	1.2	0.0046	0.030	0.0004	0.009
(2,592)			4.1	0.0046	0.030		
2×8^4	0.0004	0.004	0.9	0.0008	0.090	0.0002	0.025
(8,192)			2.9	0.0024	0.090		
2×10^4	0.0002	0.008	0.6	0.0001	0.220	5e-05	0.070
(20,000)			2.0	0.0013	0.230		
2×13^4	0.0001	0.022	0.4	0.0001	0.630	4e-06	0.178
(57,122)			1.2	0.0007	0.640		
2×14^4	5e-06	0.029	0.4	1e-05	0.860	1e-05	0.237
(76,832)			1.2	0.0005	0.880		
2×15^4	8e-06	0.036	0.4	0.0001	1.220	9e-07	0.313
(101,250)			1.2	0.0005	1.250		

We use various values of the number of points that corresponds to situations when one needs to compute the sensitivity measures with different accuracy. We have computed results for g_0 (g_0 is the integral over the integrand $g(x) = f(x) - c$, $f(x)$ is the approximate model function of UNI-DEM and c is a constant obtained as a Monte Carlo estimate of f_0, [28]), the total variance **D** as well as total sensitivity indices $S_i^{tot}, i = 1, 2, 3$. The above-mentioned parameters are presented in Table 5. Table 5 presents the results obtained for a relatively low sample size $n = 6,600$.

One can notice that for most of the sensitivity parameters, the simplest *shaking* algorithm MCA-MSS-1 outperforms the scrambled Sobol sequences, as well as the algorithm based on the $\Lambda \Pi_\tau$ Sobol sequences in terms of accuracy. For higher values of sample sizes this effect is even stronger.

One can clearly observe that the simplest *shaking* algorithm MCA-MSS-1 based on modified Sobol sequences improves the error estimates for non-smooth integrands. For smooth functions modified algorithms MCA-MSS-2 and MCA-MSS-2-S give better results than MCA-MSS-1. Even for relatively large radiuses ρ the results are good in terms of accuracy. The reason is that centers of spheres are very well uniformly distributed by definition. So that even for large values of radiuses of *shaking* the generated random points continue to be well distributed. We should stress on the fact that for relatively low number of points (< 1,000) the algorithm based on modified Sobol sequences gives results with a high accuracy.

Table 5 Relative error (in absolute value) and computational time for estimation of sensitivity indices of input parameters using various Monte Carlo and quasi-Monte Carlo approaches ($n = 6,600, c \approx 0.51365, \delta \approx 0.08$)

Estimated quantity	Sobol QMCA	Owen scrambling	MCA-MSS-1	
			ρ	Rel. error
g_0	1**e**-05	0.0001	0.0007	0.0001
			0.007	6**e**-05
D	0.0007	0.0013	0.0007	0.0003
			0.007	0.0140
S_1^{tot}	0.0036	0.0006	0.0007	0.0009
			0.007	0.0013
S_2^{tot}	0.0049	6**e**-05	0.0007	2**e**-05
			0.007	0.0034
S_3^{tot}	0.0259	0.0102	0.0007	0.0099
			0.007	0.0211

7   Conclusions

A comprehensive theoretical and experimental study of the Monte Carlo algorithm MCA-MSS-2 based on *symmetrised shaking* of Sobol sequences has been done. The algorithm combines properties of two of the best available approaches—Sobol QMC integration and a high-quality SFMT pseudorandom number generator. It has been proven that this algorithm has an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error.

A comparison with the scrambling approach, as well as with the Sobol QMC algorithm and the algorithm using SFMT generator, has been provided for numerical integration of smooth and non-smooth integrands. The algorithms mentioned above are tested numerically also for computing sensitivity measures for UNI-DEM model to study sensitivity of ozone concentration according to variation of chemical rates. All algorithms under consideration are efficient and converge with the expected rate of convergence. It is important to notice that the Monte Carlo algorithm MCA-MSS-2 based on modified Sobol sequences when *symmetrised shaking* is used has a unimprovable rate of convergence and gives reliable numerical results.

References

1. Antonov, I., Saleev, V.: An economic method of computing LP_τ-sequences. USSR Comput. Math. Math. Phys. **19**, 252–256 (1979)
2. Bradley, P., Fox, B.: Algorithm 659: implementing Sobol's quasi random sequence generator. ACM Trans. Math. Software **14**(1), 88–100 (1988)
3. Dimov, I.T.: Monte Carlo Methods for Applied Scientists. World Scientific, London (2008)
4. Dimov, I.T., Farago, I., Havasi, A., Zlatev, Z.: Operator splitting and commutativity analysis in the Danish Eulerian Model. Math. Comput. Simulat. **67**, 217–233 (2004)
5. Dimov, I.T., Ostromsky, Tz., Zlatev, Z.: Challenges in using splitting techniques for large-scale environmental modeling. In: Farago, I., Georgiev, K., Havasi, A. (eds.) Advances in Air Pollution Modeling for Environmental Security. NATO Science Series 54, pp. 115–132. Springer, Dordrecht (2005)
6. Dimov, I.T., Georgieva, R.: Monte Carlo method for numerical integration based on Sobol's sequences. In: Dimov, I., Dimova, S., Kolkovska, N. (eds.) Numerical Methods and Applications. Lecture Notes in Computer Science 6046, pp. 50–59. Springer, Berlin (2011)
7. Dimov, I.T., Georgieva, R., Ostromsky, Tz., Zlatev, Z.: Advanced algorithms for multidimensional sensitivity studies of large-scale air pollution models based on Sobol sequences. Comput. Math. Appl. **65**(3), 338–351 (2013). doi: 10.1016/j.camwa.2012.07.005
8. Faure, H., Tezuka, S.: Another random scrambling of digital (t,s)-sequences. In: (Fang, K., Hickernell, F., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods. Springer, Berlin, pp. 242–256 (2000)
9. Homma, T., Saltelli, A.: Importance measures in global sensitivity analysis of nonlinear models. Reliab. Eng. Syst. Saf. **52**, 1–17 (1996)
10. Hong, H., Hickernell, F.: Algorithm 823: implementing scrambled digital sequences. ACM Trans. Math. Software **29**(2), 95–109 (2003)
11. Joe, S., Kuo, F.: Constructing Sobol' sequences with better two-dimensional projections. SIAM J. Sci. Comput. **30**, 2635–2654 (2008)
12. Kucherenko, S., Feil, B., Shah, N., Mauntz, W.: The identification of model effective dimensions using global sensitivity analysis. Reliab. Eng. Syst. Saf. **96**, 440–449 (2011)
13. L'Ecuyer, P., Lecot, C., Tuffin, B.: A randomized quasi-Monte Carlo simulation method for Markov chains. Oper. Res. **56**(4), 958–975 (2008)
14. L'Ecuyer, P., Lemieux, C.: Recent advances in randomized quasi-Monte Carlo methods. In: Dror, M., L'Ecuyer, P., Szidarovszki, F. (eds.) Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer, Boston (2002)
15. Levitan, Y., Markovich, N., Rozin, S., Sobol, I.: On quasi-random sequences for numerical computations. USSR Comput. Math. Math. Phys. **28**(5), 755–759 (1988)
16. Marchuk, G.I.: Mathematical Modeling for the Problem of the Environment, Studies in Mathematics and Applications, No. 16. North-Holland, Amsterdam (1985)
17. Matousek, J.: On the L_2-discrepancy for anchored boxes. J. Complex. **14**, 527–556 (1998)
18. Niederreiter, H.: Low-discrepancy and low-dispersion sequences. J. Number Theor. **30**, 51–70 (1988)
19. Owen, A.: Randomly permuted (t,m,s)-nets and (t,s)-sequences. In: Niederreiter, H., Shiue, P.J.-S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics, vol. 106, pp. 299–317. Springer, New York (1995)
20. Owen, A.: Variance and discrepancy with alternative scramblings. ACM Trans. Comput. Logic. **V**, 1–16 (2002)
21. Saito, M., Matsumoto, M.: SIMD-oriented Fast Mersenne Twister: a 128-bit pseudorandom number generator. In: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 607–622. Springer, Heidelberg (2008)
22. Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M.: Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. Halsted Press, New York (2004)
23. Sobol, I.M.: Monte Carlo Numerical Methods. Nauka, Moscow (1973) (in Russian)

24. Sobol, I.M.: On the systematic search in a hypercube. SIAM J. Numer. Anal. **16**, 790–793 (1979)
25. Sobol, I.M.: On quadratic formulas for functions of several variables satisfying a general Lipschitz condition. USSR Comput. Math. Math. Phys. **29**(6), 936–941 (1989)
26. Sobol, I.M.: Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Math. Comput. Simulat. **55**(1–3), 271–280 (2001)
27. Sobol, I., Asotsky, D., Kreinin, A., Kucherenko, S.: Construction and comparison of high-dimensional Sobol' generators. Wilmott J. **2011**, 67–79 (2011)
28. Sobol, I., Myshetskaya, E.: Monte Carlo estimators for small sensitivity indices. Monte Carlo Meth. Appl. **13**(5–6), 455–465 (2007)
29. Tezuka, S.: Uniform Random Numbers, Theory and Practice. Kluwer Academic Publishers, Boston; IBM Japan, Tokyo (1995)
30. Weyl, H.: Ueber die Gleichverteilung von Zahlen mod Eins. Math. Ann. **77**(3), 313–352 (1916)
31. The NAG C Library (www.nag.co.uk/numeric/CL/CLdescription.asp.)
32. SIMD-oriented Fast Mersenne Twister (SFMT) (http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/).
33. Zlatev, Z., Dimov, I.T.: Computational and Numerical Challenges in Environmental Modelling. Elsevier, Amsterdam (2006)
34. Zlatev, Z., Dimov, I.T., Georgiev, K.: Three-dimensional version of the Danish Eulerian model. Z. Angew. Math. Mech. **76**(S4), 473–476 (1996)