Advanced Algorithms for Multidimensional Sensitivity Studies of Large-scale Air Pollution Models Based on Sobol Sequences

I. Dimov[a,*], R. Georgieva[a], Tz. Ostromsky[a], Z. Zlatev[b]

[a] *Department of Parallel Algorithms, IICT, Bulgarian Academy of Sciences, Acad. G. Bonchev 25 A, 1113 Sofia, Bulgaria*
[b] *Department of Environmental Science - Atmospheric Environment, Aarhus University, Frederiksborgvej 399, building ATMI, 4000, Roskilde, Denmark*

Abstract

In this paper advanced variance-based algorithms for global sensitivity analysis are studied. We consider efficient algorithms: Monte Carlo, quasi-Monte Carlo (QMC) and scrambled quasi-Monte Carlo algorithms based on Sobol sequences. Low discrepancy $\Lambda\Pi_\tau$ Sobol sequences are considered as a basis. Two other approaches are also analyzed. The first one is an efficient Monte Carlo (MC) algorithm for multidimensional integration based on modified Sobol sequences (MCA-MSS) and proposed in an earlier work by some of the authors [6]. Each random point in d-dimensional domain of integration is generated in the following way. A Sobol vector of dimension s ($\Lambda\Pi_\tau$ point) is considered as a centrum of a sphere with a radius ρ. Then a random point uniformly distributed on the sphere is taken and a random variable is defined as a value of the integrand at that random point. It is proved that the Monte Carlo algorithm based on Sobol sequences MCA-MSS has an optimal rate of convergence for functions with continuous and bounded first derivatives in terms of probability and mean square error. The second one is a randomized

*Corresponding author
Email addresses: `ivdimov@bas.bg` (I. Dimov), `rayna@parallel.bas.bg` (R. Georgieva), `ceco@parallel.bas.bg` (Tz. Ostromsky), `zz@dmu.dk` (Z. Zlatev)
URL: `http://parallel.bas.bg/dpa/BG/dimov/index.html` (I. Dimov), `http://parallel.bas.bg/~rayna/` (R. Georgieva), `http://parallel.bas.bg/~ceco/` (Tz. Ostromsky), `http://www.dmu.dk/AtmosphericEnvironment/staff/zlatev.htm` (Z. Zlatev)

QMC algorithm proposed by Art Owen [19]. The procedure of randomization in the latter case is also known as *Owen scrambling.*

The chosen algorithms are applied to sensitivity studies of air pollution levels calculated by the Unified Danish Eulerian Model (UNI-DEM) to some chemical rate reactions. UNI-DEM is chosen as a case-study since it is a typical large-scale mathematical model in which the chemical reactions are strongly and adequately presented. Large number of numerical experiments is performed. Conclusions about applicability and efficiency of the algorithms under consideration are drawn.

Keywords: Monte Carlo method, multidimensional integration, Sobol sequences, Scrambled Sobol sequences
2010 MSC: 60H35, 65R10,65R99

---

1. Introduction

Sensitivity analysis (SA) is an important issue for large-scale mathematical models. There are several available sensitivity analysis techniques [25]. 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 input factor and model output). Among quantitative methods, variance-based methods are the most often used [24]. The idea of these methods is to estimate how the variation of an input parameter or a group of inputs contributes into the variance of model output. A careful sensitivity analysis is needed in order to decide where and how simplifications of the model can be made. As a measure of this analysis we use the *total sensitivity indices (TSI)* (see, Section 2). We find this measure more adequate and reliable then other measures [7] when multi-component analysis is needed. This is very important when one deals with very large and *computationally heavy* mathematical models.

The focus of our study is in the area of environmental security (air pollution transport). Contemporary mathematical models of air pollution transport should include a fairly large set of chemical and photochemical reactions to be established as a reliable simulation tool [40]. The investigations and the numerical results reported in this paper have been done by using a large-scale mathematical model called the Unified Danish Eulerian Model [38, 39, 41].

It is important to analyze the influence of variations of the chemical rates on the model results in order to make right assumptions about the simplifications which have to be implemented. Such an analysis can give valuable information about the performance of reliable and reasonable simplifications or to identify parameters and mechanisms the accuracy of which should be improved, because the model results are very sensitive to variations of these parameters and mechanisms. 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.

Since TSI are considered as main measure of sensitivity it is important to have efficient algorithms for estimating TSI. As it is shown in Section 2 the total indices may be presented as multidimensional integrals:

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

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

Some applications lead to high-dimensional integration. That's why efficient numerical methods for high-dimensional integration are strongly appreciated.

2. Problem Setting

2.1. Modeling and Sensitivity

It is assumed that the mathematical model can be presented as a model function

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

is a vector of input parameters with a joint **p**robability **d**ensity **f**unction (p.d.f.) $p(\mathrm{x}) = p(x_1, \ldots, x_d)$. It is also assumed that input variables are independent (non-correlated input variables) and the density function $p(\mathrm{x}) = p(x_1, x_2, \ldots, x_d)$ is known, even if x_i are not actually random variables. This implies that the output u is also a random variable, as it is a function of the random vector x, with its own p.d.f. The above presentation is quite general. One has to take into account that in most cases of large-scale modeling the function $f(\mathrm{x})$ is not explicitly known. Very often the function $f(\mathrm{x})$ is a solution of a system of partial differential equations with boundary and initial conditions. The latter conditions have to ensure the existence of a unique solution of the system. Another point is that for large-scale problems containing differential equations with many parameters is not easy to prove existence of a unique solution. Often people assume that the unique solution exists and then they develop discretization methods. Each method may have more than one algorithm for its implementation. The choice of the *most efficient algorithm* is of great importance for large-scale problems. Obviously, the most efficient algorithm is the algorithm that implements a given method and solves the problem with a fixed accuracy and *smallest computational complexity*. The computational complexity is considered with respect to the number of arithmetic operations needed to solve the problem with a given accuracy (given error, in case of deterministic algorithms, or given probability error in case of stochastic algorithms)[1]. In such a way the whole set of differential equations with associated initial and boundary conditions, numerical methods for discretization, as well as algorithms that implement numerical methods is considered as a *mathematical model*. We stress on the fact that after discretization some large-scale mathematical models may contain billions or trillions of algebraic equations. To run such a model one needs many days (even months) to get the solution even when high-performance computers are used. That's why the development of efficient algorithms is an important issue. At the same time the large-scale models describe both - *important*, as well as *not very important* phenomenon from the point of view of the main output of the model. Both sets of phenomenon are treated equally. It may happen that the treatment of a not important phenomenon in the model may take large amount of computational resources when the contribution of this phenomenon to the solution is smaller than the computational error. It also may happen that the output of the model is quite sensitive to a single input measurable parameter. Such an information may help in the following way: the above parameter may be measured with a higher accuracy (it means, by investing money to buy more accurate instruments to measure such a parameter). That's why it is reasonable to introduce an indicator that measures the importance of the influence of a given input parameter onto the output. The main indicator referred to a given input parameter

[1]There are some considerations in complexity analysis when both arithmetic and logical operations are taken into account with some weights [4]; sometimes, the volume of needed operational memory and the cost of communications are taken into account.

$x_i,\ i = 1, \ldots, d$ (normalised between 0 and 1) is defined as

$$\frac{\mathbf{D}[\mathbf{E}(\mathrm{u}|x_i)]}{\mathbf{D}_\mathrm{u}}, \tag{3}$$

where $\mathbf{D}[\mathbf{E}(\mathrm{u}|x_i)]$ is the variance of the conditional expectation of u with respect to x_i and $\mathbf{D}_\mathrm{u}$ is the total variance according to u. The *total sensitivity index* [11] provides a measure of the total effect of a given parameter, including all the possible joint terms between that parameter and all the others. The **t**otal **s**ensitivity **i**ndex (TSI) of input parameter $x_i, i \in \{1, \ldots, d\}$ is defined in the following way [11, 31]:

$$S_{x_i}^{tot} = S_i + \sum_{l_1 \neq i} S_{il_1} + \sum_{l_1, l_2 \neq i, l_1 < l_2} S_{il_1 l_2} + \ldots + S_{il_1 \ldots l_{d-1}}, \tag{4}$$

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 $j^{\text{-th}}$ order sensitivity index. The higher-order terms describe the interaction effects between the unknown input parameters $x_{i_1}, \ldots, x_{i_\nu}, \nu \in \{2, \ldots, d\}$ on the output variance. Later on we will show how sensitivity indices $S_{l_1\ \ldots\ l_\nu}$ are defined via the variances of conditional expectations $\mathbf{D}_{l_1} = \mathbf{D}[f_{l_1}(x_{l_1})] = \mathbf{D}[\mathbf{E}(\mathrm{u}|x_{l_1})]$, $\mathbf{D}_{l_1\ \ldots\ l_\nu}, 2 \leq \nu \leq d$.

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:

$$f(\mathrm{x}) = f_0 + \sum_{\nu=1}^{d} \sum_{l_1 < \ldots < l_\nu} f_{l_1 \ldots l_\nu}(x_{l_1}, x_{l_2}, \ldots, x_{l_\nu}), \tag{5}$$

where f_0 is a constant. The total number of summands in equation (5) is 2^d. The representation (5) is called ANOVA-representation of the model function $f(\mathrm{x})$ if each term is chosen to satisfy the following condition

$$\int_0^1 f_{l_1 \ldots l_\nu}(x_{l_1}, x_{l_2}, \ldots, x_{l_\nu}) \mathrm{d}x_{l_k} = 0, \quad 1 \leq k \leq \nu, \quad \nu = 1, \ldots, d.$$

Let us mention the fact that if the whole presentation (5) of the right-hand site is used, then it doesn't simplify the problem. The hope is that a truncated sequence $\sum_{\nu=1}^{d_{tr}} \sum_{l_1 < \ldots < l_\nu} f_{l_1 \ldots l_\nu}(x_{l_1}, x_{l_2}, \ldots, x_{l_\nu})$, where $d_{tr} < d$ can be considered as a good approximation to the model function f.

The quantities

$$\mathbf{D} = \int_{U^d} f^2(\mathrm{x})\mathrm{dx} - f_0^2, \quad \mathbf{D}_{l_1\ \ldots\ l_\nu} = \int f_{l_1\ \ldots\ l_\nu}^2 \mathrm{d}x_{l_1} \ldots \mathrm{d}x_{l_\nu} \tag{6}$$

are called total and partial variances respectively and have been obtained after squaring and integrating over U^d the equality (5) on the assumption that $f(\mathrm{x})$ is a square integrable function (thus all terms in (5) are also square integrable functions). Therefore, the total variance of the model output is partitioned into partial variances in the analogous way as the model function, that is the unique ANOVA-decomposition: $\mathbf{D} = \sum_{\nu=1}^{d} \sum_{l_1<\ldots<l_\nu} \mathbf{D}_{l_1 \ldots l_\nu}$. It is obvious that the use of terms of probability theory 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 (6). For example f_{l_1} is presented by a conditional expectation:

$$f_{l_1}(x_{l_1}) = \mathbf{E}(\mathrm{u}|x_{l_1}) - f_0 \quad \text{and respectively} \quad \mathbf{D}_{l_1} = \mathbf{D}[f_{l_1}(x_{l_1})] = \mathbf{D}[\mathbf{E}(\mathrm{u}|x_{l_1})].$$

Based on the above assumptions about the model function and the output variance, the following quantities $S_{l_1\ \ldots\ l_\nu} = \dfrac{\mathbf{D}_{l_1\ \ldots\ l_\nu}}{\mathbf{D}}, \quad \nu \in \{1, \ldots, d\}$ are called global sensitivity indices [32]. This formula coincides for $\nu = 1$ with (3) and the so defined measures correspond to the main effect of input parameters as well as the interactions effect.

Based on the results discussed above it is clear that the mathematical treatment of the problem of providing global sensitivity analysis consists in evaluating total sensitivity indices (4) of corresponding order. And that leads to computing of multidimensional integrals (1). It means that in general case one needs to compute 2^d integrals of type (6) to obtain $S_{x_i}^{tot}$. As we discussed earlier the basic assumption underlying representation (5) is that the basic features of the model functions (2) describing typical real-life problems can be presented by low-order subsets of input variables containing, terms of the order up to d_{tr}, where $d_{tr} < d$. Therefore, based on this assumption, one can assume that the dimension of the initial problem can be reduced. Following Sobol [32] we consider an arbitrary set of m variables $(1 \leq m \leq d-1)$: $\mathrm{y} = (x_{k_1}, \ldots, x_{k_m}),\ 1 \leq k_1 < \ldots < k_m \leq d,$ and let z be the set of $d - m$ complementary variables. Thus $\mathrm{x} = (\mathrm{y}, \mathrm{z})$. Let $K = (k_1, \ldots, k_m)$.

The variances corresponding to the subsets y and z can be defined as

$$\mathbf{D}_{\mathrm{y}} = \sum_{n=1}^{m} \sum_{(i_1<\ldots<i_n)\in K} \mathbf{D}_{i_1\ \ldots\ i_n}, \qquad \mathbf{D}_{\mathrm{z}} = \sum_{n=1}^{d-m} \sum_{(j_1<\ldots<j_n)\in \bar{K}} \mathbf{D}_{j_1\ \ldots\ j_n}, \tag{7}$$

where the complement of the subset K in the set of all parameter indices is denoted by $\bar{K}$. The first sum in (7) is extended over all subsets $(i_1, \ldots, i_n)$, where all indices $i_1, \ldots, i_n$ belong to K. Then the total variance corresponding to the subset y is $\mathbf{D}_{\mathrm{y}}^{tot} = \mathbf{D} - \mathbf{D}_{\mathrm{z}}$ and it is extended over all subsets $(i_1, \ldots, i_\nu)$, $1 \leq \nu \leq d$, where at least one $i_l \in K$, $1 \leq l \leq \nu$.

The procedure for computation of global sensitivity indices is based on the following representation of the variance $\mathbf{D}_{\mathrm{y}}$: $\mathbf{D}_{\mathrm{y}} = \int f(\mathrm{x})\, f(\mathrm{y}, \mathrm{z}')\mathrm{dx}\mathrm{dz}' - f_0^2$ (see [32]). The last equality allows to construct a Monte Carlo algorithms for evaluating $f_0, \mathbf{D}$ and $\mathbf{D}_{\mathrm{y}}$, where $\xi = (\eta, \zeta)$:

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

For example, for $m = 1, \mathrm{y} = \{x_{l_1}\}, l_1 \in \{1, \ldots, d\}$ and $\mathrm{z} = \{1, \ldots, d\} \backslash l_1$: $S_{l_1} = S_{(l_1)} = \mathbf{D}_{(l_1)}/\mathbf{D}$, $S_{l_1}^{tot} = \mathbf{D}_{l_1}^{tot}/\mathbf{D} = 1 - S_{\mathrm{z}}$.

The computing of higher-order interactions effect can be performed by an iterative process. For example, $S_{(l_1 l_2)} = \mathbf{D}_{(l_1 l_2)}/\mathbf{D} = S_{l_1} + S_{l_2} + S_{l_1 l_2}$, and $S_{l_1 l_2}$ can be obtained assuming that the corresponding first-order sensitivity indices have been already computed.

Instead of randomized (Monte Carlo) algorithms for computing the above sensitivity parameters one can try deterministic quasi-Monte Carlo algorithms, or randomized quasi-Monte Carlo. Randomized (Monte Carlo) algorithms have proven to be very efficient in solving multidimensional integrals in composite domains [4, 27]. 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 dimensions (up to $d = 10$) [30]. Sobol $\Lambda\Pi_\tau$ sequences are good candidates for efficient QMC algorithms. Algorithms based on $\Lambda\Pi_\tau$ sequences mimic the pseudo-random sequences used in Monte Carlo integration, but actually they are deterministic. One of the problems with $\Lambda\Pi_\tau$ sequences is that they may have *bad* two-dimensional projection. To overcome this problem randomized QMC can be used. There are several ways of randomization and the *scrambling* is one of them. The original motivation of scrambling [12, 19] aims toward obtaining more uniformity for quasi-random sequences in high dimensions, which can be checked via two-dimensional projections.

Owen scrambling [19], called nested scrambling, was developed to provide a practical error estimate for QMC based on treating each scrambled sequence as a different and independent random sample from a family of randomly scrambled quasi-random numbers. Actually, the scrambled algorithms may be considered as Monte Carlo algorithms with a special choice of the density function. It's a matter of definition. Thus, there are two classes of algorithms we have to compare: *deterministic* and *randomized.*

3. Complexity in Classes of Algorithms

We need to find a setting in which one can consider and compare two classes of algorithms: *deterministic algorithms* and *randomized (Monte Carlo) algorithms.* Usually randomized algorithms reduce problems to the approximate calculation of mathematical expectations. We use the following notation. The mathematical expectation of the random variable (r.v.) θ is denoted by $E_\mu(\theta)$ (sometimes abbreviated to $E\theta$). By $x = (x_1, \dots, x_d)$ we denote a point in a closed domain $\Omega \subset \mathbf{R}^d$, where $\mathbf{R}^d$ is d-dimensional Euclidean space. The d-dimensional unit cube is denoted by $E^d = [0,1]^d$. We shall further denote the values (realizations) of a random point ξ or r.v. θ by $\xi^{(i)}$ and $\theta^{(i)} (i = 1, 2, \dots, n)$ respectively. If $\xi^{(i)}$ is a d-dimensional random point, then usually it is constructed using d random numbers γ, i.e., $\xi^{(i)} \equiv (\gamma_{i,1}, \dots, \gamma_{i,d})$. Let I be the desired value of the integral. Assume for a given r.v. θ one can prove that $E\theta = I$. Suppose the mean value of n values of θ: $\theta^{(i)}$, $i = 1, \dots, n$ is considered as a Monte Carlo approximation to the solution: $\bar{\theta}_n = 1/n \sum_{i=1}^{n} \theta^{(i)} \approx I$. One can only state that 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 (even high) probability. In most cases of practical computations it is reasonable to accept an error estimate with a probability smaller than 1.

Consider the following problem of integration:

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

where $x \equiv (x_1, \dots, x_d) \in E^d \subset \mathbf{R}^d$ and $f \in C(E^d)$ is an integrable function on E^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 $S(f) = \int_{E^d} f(x)dx$ and $f \in F_0 \subset C(E^d)$. We will call S 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 (or approximate) $S(f)$. One may be interested to consider 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 $F_0 \equiv \mathbf{W}^k(\|f\|; \mathbf{E}^d)$:

Definition 3.1. *Let d and k be integers, and $d, k \geq 1$. We consider the class $\mathbf{W}^k(\|f\|; \mathbf{E}^d)$ (sometimes abbreviated to $\mathbf{W}^k$) of real functions f defined over the unit cube $\mathbf{E}^d = [0,1)^d$, possessing all the partial derivatives*

$$\frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}, \quad \alpha_1 + \ldots + \alpha_d = r \leq k,$$

which are continuous when $r < k$ and bounded in sup *norm when $r = k$. The semi-norm $\|\cdot\|$ on $\mathbf{W}^k$ is defined as*

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

We will call a *quadrature formula* any expression

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

which approximates the value of the integral $S(f)$. The real numbers $c_i \in \mathbf{R}$ are called weights and d dimensional points $x^{(i)} \in E^d$ are called nodes. It is clear that for fixed weights c_i and nodes $x^{(i)} \equiv (x_{i,1}, \ldots x_{i,d})$ the quadrature formula $A(f, n)$ may be used to define an algorithm. 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) algorithms $\mathcal{A}^{\mathcal{R}}$.

Definition 3.2. *Given a randomized (Monte Carlo) integration formula for the functions from the space $\mathbf{W}^k$ by $err(f, A^R)$ we denote the integration error*

$$\int_{\mathbf{E}^d} f(x)dx - A^R(f, n),$$

by $\varepsilon_p(f)$ the probability error meaning that $\varepsilon_p(f)$ is the least possible real number with

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

and by $r(f)$ the mean square error

$$r(f) = \left\{ E\left[\int_{\mathbf{E}^d} f(x)dx - A^R(f,n)\right]^2 \right\}^{1/2}.$$

We assume that one is happy 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 one may use $\varepsilon_P(f)$ as an accuracy measure for both randomized and deterministic algorithms. In such a way it is consistent to consider a wider class $\mathcal{A}$ of algorithms that contains both classes: randomized and deterministic algorithms.

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

$$\mathcal{A} = \{A : Pr(\varepsilon_p(f) \leq \varepsilon) \geq c\}$$

that solve a given problem with a probability error $\varepsilon_p(f)$ such that the probability that $\varepsilon_p(f)$ is less than a priori given constant ε is bigger than a constant $c < 1$.

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 [35]. For practical purposes an u.d.s. must be found that satisfied three requirements [27, 29]:

(i) the best asymptote as $n \to \infty$;

(ii) well distributed points for small n;

(iii) a computationally inexpensive algorithm.

All $\Lambda\Pi_\tau$-sequences given in [29] satisfy the first requirement. Good distributions like $\Lambda\Pi_\tau$ sequences are also called (t, m, s)-nets and (t, s)-sequences in base b. To introduce them, define first an elementary s-interval in base b as a subset of $\mathbf{E}^s$ of the form $\prod_{j=0}^{s} \left[\frac{a_j}{b^{d_j}}, \frac{a_j+1}{b^{d_j}}\right]$, where a_j, d_j are integers and $a_j < d_j$ for all $j \in \{1, ..., s\}$. Given 2 integers $0 \leq t \leq m$, a (t, m, s)-net in base b is a sequence $x^{(i)}$ of b^m points of $\mathbf{E}^s$ such that $Card\ P \cap \{x^{(1)}, \ldots, x^{(b^m)}\} = b^t$ for any elementary interval P in base b of hypervolume $\lambda(P) = b^{t-m}$.

Given a non-negative integer t, a (t, s)-sequence in base b is an infinite sequence of points $x^{(i)}$ such that for all integers $k \geq 0, m \geq t$, the sequence $\{x^{(kb^m)}, \ldots, x^{((k+1)b^m-1)}\}$ is a (t, m, s)-net in base b.

I.M. Sobol 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 H. 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 $m_{k,j}, 1 \leq k \leq s_j$, is odd and less than 2^k. 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, 28]. The work [15] contains more details.

4.2. The Monte Carlo Algorithm based on Modified Sobol Sequences - MCA-MSS

The algorithm was proposed recently by the first two authors of this paper in [6]. The general 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

ρ. A random point $\xi \in \mathbf{E}^d$ 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 $\xi^{(i)}(\rho) \in \mathbf{E}^d$. Assume $\xi^{(i)}(\rho) = x^{(i)} + \rho\omega^{(i)}$, where ρ is relatively small $\rho << \left[\frac{a_j}{2^{d_j}}, \frac{a_j+1}{2^{d_j}}\right]$, such that $\xi^{(i)}(\rho)$ is still in the same elementary j-interval $\mathbf{E}_i^d = \prod_{j=0}^{d}\left[\frac{a_j}{2^{d_j}}, \frac{a_j+1}{2^{d_j}}\right]$, where the pattern $\Lambda\Pi_\tau$ point $x^{(i)}$ is. We use a subscript i in $\mathbf{E}_i^d$ to indicate that the i-th $\Lambda\Pi_\tau$ point $x^{(i)}$ is in it. So, we assume that if $x^{(i)} \in \mathbf{E}_i^d$, then $\xi^{(i)}(\rho) \in \mathbf{E}_i^d$ too.

In [6] was proven that the mathematical expectation of the random variable $\theta = f(\xi)$ is equal to the value of the integral (8), that is

$$E\theta = S(f) = \int_{E^d} f(x)dx.$$

This result allows to define a randomized algorithm. One can take the Sobol $\Lambda\Pi_\tau$ point $x^{(i)}$ and *shake* it a little bit. *Shaking* means to define random points $\xi^{(i)}(\rho) = x^{(i)} + \rho\omega^{(i)}$ according to the procedure described above.

Let us now analyse the probability error of the algorithm. Let us assume that $n = m^d$, $m \geq 1$. We divide the unit cube $\mathbf{E}^d$ into m^d disjoint cubes, such that they coincide with the elementary sub-cubes (or subintervals) defined in Subsection 4.1 $\mathbf{E}^d = \bigcup_{j=1}^{m^d} K_j$, where $K_j = \prod_{i=1}^{d}[a_i^j, b_i^j)$, with $b_i^j - a_i^j = \frac{1}{n}$ for all $i = 1, \dots, d$.

In such a way in each d-dimensional sub-cube K_i there is exactly one $\Lambda\Pi_\tau$ point $x^{(i)}$. Assuming that after shaking, the random point stays inside K_i, i.e. $\xi^{(i)}(\rho) = x^{(i)} + \rho\omega^{(i)} \in K_i$ one may try to exploit the smoothness of the integrand. Indeed, let us consider the case when the integrand has continuous and bounded first derivatives: $f \in F_0 \equiv \mathbf{W}^1(L; \mathbf{E}^d)$, where $L = \|f\|$.

Then, if $p(x)$ is a probability density function, such that $\int_{E^d} p(x)dx = 1$, then

$$\int_{K_j} p(x)dx = p_j \leq \frac{c_1}{n},$$

where c_1 is a constant. If d_j is the diameter of K_j, then

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

where c_2 is another constant.

In the particular case when the subintervals are sub-cubes with edge $1/m$ we have: $c_1 = 1$ and $c_2 = \sqrt{d}$.

Suppose according to our procedure, we select a random points $\xi^{(j)}$ from each cube K_j, and calculate all function values $f(\xi^{(j)}), j = 1, \dots, m^d$.

We approximate the value of the integral in the following way:

$$I(f) \approx \frac{1}{m^d} \sum_{j=1}^{n} f(\xi(x^{(j)}).$$

We prove the following

Theorem 4.1. *The quadrature formula constructed above satisfies*

$$\varepsilon(f,d) \le c_d^{'} \|f\| \, n^{-\frac{1}{2}-\frac{1}{d}}$$

and

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

where the constants $c_d^{'}$ and $c_d^{''}$ do not depend on n.

Proof 4.1. *One can see that*

$$E \left\{ \frac{1}{m^d} \sum_{j=1}^{n} f(\xi(x^{(j)})) \right\} = \int_{\mathbf{E}^d} f(x) dx.$$

For the fixed $\Lambda\Pi_\tau$ point $x^{(j)} \in K_j$ we have:

$$f(\xi^{(j)}) = f(x^{(j)}) + \sum_{s=1}^{d} f'(\eta_s^j)(\xi_s^{(j)}) - x_s^{(j)}),$$

where $\eta^j \in K_j$.

Since $f \in \mathbf{W}^1(L; \mathbf{E}^d)$, $f'(\eta_s^j) \le L_j$ and $L = \|f\|$ is the majorant for all L_j, i.e. $L_j \le L$ for $j = 1, \dots, d$.

Obviously, we have

$$\begin{aligned} D f(\xi^{(j)}) &\le E f^2(\xi^{(j)}) \le L_j^2 E(\xi_s^{(j)}) - x_s^{(j)})^2 \\ \le \;\; & L_j^2 \sup_{x_1^{(j)}, x_2^{(j)}} \left| x_1^{(j)} - x_2^{(j)} \right|^2 \le L_j^2 (c_2^{(j)})^2 n^{-2/d}. \end{aligned}$$

Now the variance $D\theta_n$ can be estimated:

$$D\theta_n = \sum_{j=1}^{n} p_j^2 Df(\xi^{(j)}) \leq \sum_{j=1}^{n} ((c_1^{(j)})^2 n^{-2} L_j^2 (c_2^{(j)})^2 n^{-2/d}$$

$$\leq \left(L_j c_1^{(j)} c_2^{(j)}\right)^2 n^{-1-2/d}. \quad (9)$$

Therefore

$$r(f,d) \leq c_d^{''} \|f\| n^{-\frac{1}{2}-\frac{1}{d}}.$$

The application of the Tchebychev's inequality to the variance (9) yields

$$\varepsilon(f,d) \leq c_d^{'} \|f\| n^{-\frac{1}{2}-\frac{k}{d}}$$

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

One can see that the Monte Carlo algorithm based on Sobol sequences MCA-MSS has an optimal rate of convergence for functions with continuous and bounded first derivative [4]. This means that the rate of convergence ($n^{-\frac{1}{2}-\frac{1}{d}}$) can not be improved for the functional class $\mathbf{W}^1$ in the class of the randomized algorithms $\mathcal{A}^{\mathcal{R}}$.

4.3. Owen Nested Scrambling Algorithm

Owen scrambling [19], called nested scrambling, was developed to provide a practical error estimate for QMC based on treating each scrambled sequence as a different and independent random sample from a family of randomly scrambled quasi-random numbers. Moreover, scrambling gives us a simple and unified way to generate quasi-random numbers for parallel, distributed, and grid-based computational environments.

After Niederreiter sequences [18] were proposed, Owen [19] and Tezuka [34] in 1994 independently developed two powerful scrambling methods for (t, s)-sequences. Owen also explicitly pointed out that scrambling can be used to provide error estimates for QMC. Although many other methods for scrambling (t, s)-sequences have been proposed, most of them are really modified or simplified Owen and Tezuka schemes. Owen scheme is theoretically powerful for (t, s)-sequences. Tezuka algorithm was proved to be efficient for $(0, s)$-sequences. Most proposed scrambling methods randomize a single digit at a time. In contrast, the scheme proposed in [3] randomizes many digits in a single point at a time, and is very efficient when using standard pseudo-random number generators as scrambler.

The idea of Owen nested scrambling is based on randomization of a single digit at each iteration. Let $x_n = (x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(s)})$ be a quasi-random number in $[0,1)^s$, and let $z_n = (z_n^{(1)}, z_n^{(2)}, \ldots, x_n^{(s)})$ be the scrambled version of the point x_n. Suppose each $x_n^{(j)}$ can be represented in base b as $x_n^{(j)} = (0.x_{n1}^{(j)}\, x_{n2}^{(j)} \ldots x_{nK}^{(j)} \ldots)_b$ with K being the number of digits to be scrambled. Then nested scrambling proposed by Owen [19, 21] can be defined as follows: $z_{n1}^{(j)} = \pi_\bullet(x_{n1}^{(j)})$, and $z_{ni}^{(j)} = \pi_{\bullet x_{n1}^{(j)} x_{n2}^{(j)} \ldots x_{ni-1}^{(j)}}(x_{ni}^{(j)})$, with independent permutations $\pi_{\bullet x_{n1}^{(j)} x_{n2}^{(j)} \ldots x_{ni-1}^{(j)}}$ for $i \geq 2$. Of course, (t,m,s)-net remains (t,m,s)-net under nested scrambling. However, nested scrambling requires b^{i-1} permutations to scramble the i-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 requires a considerable amount of bookkeeping, and leads 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_i's. These include Owen nested scrambling [19, 21], Tezuka's generalized Faure sequences [34], and Matousek's linear scrambling [17].

The rate for scrambled net Monte Carlo is $n^{-3/2}(log\ n)^{(s-1)/2}$ in probability while the rate for unscrambled nets is $n^{-1}(log\ n)^{s-1}$ or $n^{-1}(log\ n)^s$ along (t,s) sequences [20]. The first rate is an average case result for a fixed function f, taken over random permutations. The other results describe the worst case over functions, for a fixed set of integration points. Because scrambled nets remain nets, the worst-case bounds also apply to them [20].

Certain scrambling techniques do not affect the asymptotic discrepancy of these sequences [19]. Although scrambled quasi-random sequences improve the quality of quasi-random sequences, that improvement cannot be seen directly in the calculation of L_2 discrepancy.

Although scrambling does not change the theoretical bounds on discrepancy of these sequences, scrambling methods do improve the measures of two-dimensional projections and evaluation of high-dimensional integrals. In addition, theoretically it is impossible to prove that one of scrambled quasi-random sequences has better performance than the others so far.

5. Case-study: Variance-based Sensitivity Analysis of the Unified Danish Eulerian Model

The input data for sensitivity analysis has been obtained during runs of a large-scale mathematical model for remote transport of air pollutants (**Uni**fied **D**anish **E**ulerian **M**odel, UNI-DEM, [38, 39]). The model gives the possibility to study concentration variations in time of a high number of air pollutants and other species over a large geographical region (4800 $\times$ 4800 km), covering the whole of Europe, the Mediterranean and some parts of Asia and Africa which is important for environmental protection, agriculture, health care. It takes into account the main physical, chemical and photochemical processes between the studied species, the emissions, the quickly changing meteorological conditions. Both non-linearity and stiffness of the equations are mainly introduced by the chemistry [39]. 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 [5, 8]. 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 the 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 input parameters and the concentrations of pollutants to be output parameters.

6. Numerical Results and Observations

A number of numerical experiments have been performed to study numerically properties of the algorithms. Our expectations based on theoretical results and a large number of numerical experiments are that for non-smooth functions our Monte Carlo algorithm outperforms the QMC even for relatively low dimensions. It was interesting to see how behave the randomized QMC based on scrambled Sobol sequences.

Here we present some tests run for the following non-smooth integrand:

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

for which even the first derivative does not exist. Applications like that appear 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:

$$f_2(x_1, x_2, x_3, x_4) = x_1\ x_2^2\ \mathbf{e}^{x_1 x_2} \sin x_3 \cos x_4. \tag{10}$$

The second integrand (10) is an infinitely smooth function with a referent value of the integral $S(f_2)$ approximately equal to 0.10897. The integration domain in both cases is $E^4 = [0, 1]^4$. The results from the numerical integration tests with non-smooth and smooth integrand are presented in Table 1 and Table 2 respectively. Relative error (defined as the absolute error divided by the referent value) and computational time are chosen to be major numerical indicators of algorithm efficiency.

In this work the algorithm with Gray code implementation [1] and sets of direction numbers proposed by Joe and Kuo [10] for generating Sobol quasi-random sequences are used. Our Monte Carlo algorithm (MCA) [6] involves generating random points uniformly distributed on a sphere with a radius ρ. One of the best available random number generators, SIMD-oriented Fast Mersenne Twister (SFMT) [22, 37] 128-bit pseudo-random number generator of period $2^{19937} - 1$ has been used to generate the required random points. The radius ρ depends on the integration domain, number of samples and minimal distance between Sobol deterministic points δ. On the other hand, we observed experimentally that the behaviour 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 [36]. This implementation of

Table 1: Relative error and computational time for numerical integration of a non-smooth function ($S(f_1) \approx 7.22261$).

n	SFMT		Sobol quasi MC algorithm		Scrambled Sobol sequences		MCA-MSS		
	Rel. error	Time (s)	Rel. error	Time (s)	Rel. error	Time (s)	ρ $\times 10^3$	Rel. error	Time (s)
10^2	0.0040	0.001	0.0037	0.001	0.0214	0.001	3.9	0.0069	0.001
							13	0.0026	0.001
5.10^2	0.0007	0.005	0.0032	0.001	8e-05	0.001	2.1	0.0006	0.010
							6.9	0.0001	0.011
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
3.10^3	0.0005	0.030	0.0016	0.005	8e-05	0.005	1.2	0.0008	0.037
							4.1	0.0008	0.043
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
10^4	0.0012	0.102	0.0004	0.012	0.0006	0.014	0.8	8e-05	0.145
							2.8	0.0002	0.148
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

Table 2: Relative error and computational time for numerical integration of a smooth function ($S(f_2) \approx 0.10897$).

n	SFMT		Sobol quasi MC algorithm		Scrambled Sobol sequences		MCA-MSS		
	Rel. error	Time (s)	Rel. error	Time (s)	Rel. error	Time (s)	ρ $\times 10^3$	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
3.10^4	0.0032	0.047	6e-05	0.005	0.0002	0.006	0.6	0.0004	0.210
							1.9	0.0008	0.215
5.10^4	0.0032	0.082	9e-05	0.009	5e-05	0.007	0.4	2e-05	0.330
							1.4	0.0006	0.340

scrambled quasi-random sequences is based on TOMS Algorithm 823 [12]. 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 based on the method of Faure and Tezuka [9], and the last method combines the first two (it is referred as a *combined* approach).

Let us briefly discuss the information shown on Table 1 and Table 2. The results obtained using SFMT pseudo-random number generator have been denoted by *SFMT*. The results obtained using Sobol quasi-random sequences - by *Sobol quasi MC algorithm*; the results for scrambled Sobol quasi-random sequences - by *Scrambled Sobol sequences*, as well as the results corresponding to the Monte Carlo algorithm have been denoted by *MCA-MSS*.

Table 1 contains the results computed for the relative error (in absolute value) and the computational time for different radius coefficients κ ($\kappa = 0.03$ and $\kappa = 0.1$) for numerical integration of the non-smooth integrand f_1. For the smooth integrand f_2 the same parameters are presented in Table 2. Numerical tests with all the above mentioned algorithms have been provided in both cases of smooth and non-smooth integrand. Comparing results presented on Table 1 and Table 2 one may observe that

- trends are similar - the relative error decreases with an increase of number of points;
- all three algorithms are efficient and converge with the expected rate of convergence;
- SFMT is better for relatively small samples (n) than the Sobol algorithm in case of non-smooth function, which was expected;
- in case of non-smooth function our Monte Carlo algorithm MCA-MSS gives similar results as scrambled QMC; for several values of n we can observe advantages for MCA-MSS;
- both MCA-MSS and scrambled QMC are better in case of non-smooth functions;
- in case of smooth functions Sobol algorithm is better than SFMT, but the scrambled QMC and MCA-MSS are much better than the classical Sobol algorithm; in many cases our MCA-MSS gives a higher accuracy than the scrambled algorithm.

New random points for our algorithm have been generated using original Sobol sequences and modeling a random direction in d-dimensional space. The computational time of the calculations with pseudo-random numbers generated by SFMT (see columns labeled as *SFMT* and *MCA-MSS* in Table 1 and Table 2) has been estimated for all 10 algorithm runs. In the case of modified Sobol points when a new generated pseudo-random point is outside the integration domain, this point is rejected and a new random direction is chosen while the new random point gets into the domain. For this reason an additional computational time is needed to generate random points inside the integration domain. The computational time needed for generating the original Sobol sequences and computing the minimal distance between $\Lambda\Pi_\tau$ points in $\mathbf{E}^d$ is not included in the total time of the corresponding Monte Carlo algorithm since the same quasi-random points can be used for any integrand. The total computational complexity of the algorithm based on modified Sobol sequences does not increase essentially when the number of points n is not too large in comparison with the case when Sobol sequences are used. For large values of n the complexity increases mainly because of the algorithm for finding the minimal distance between $\Lambda\Pi_\tau$ points in $\mathbf{E}^s$. This algorithm requires $O(n^2)$ operations.

The algorithm based on modified Sobol sequences follows the relative error tendency of the original algorithm as one can expect. The numerical tests show certain advantages of this Monte Carlo algorithm according to the estimated error in comparison to the original one, as well as to the algorithm using pseudo-random numbers generated by SFMT generator.

One may also observe that for the integrands chosen here the scrambling algorithm do not outperform the algorithm with the original Sobol points, but the scrambled algorithm and Monte Carlo algorithm MCA-MSS are more stable with respect to relative errors for small values of n.

There is not universal pseudo-random number generators and it is reasonable to find the most proper generator for a given problem; for example, the convergence rate in numerical tests with the smooth integrand (10) is comparatively slow - we do not obtain an essential improvement of the relative error even for 70000 points.

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 4-th degree with 35 unknown coefficients is used for data approximation of the *mesh functions* containing model outputs as it is described in our previous paper [7].

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, [33]), as well as the total variance $\mathbf{D}$, first order sensitivity indices $S_i, i = 1, 2, 3$, and total sensitivity indices $S_i^{tot}, i = 1, 2, 3$. The above mentioned parameters are presented in Table 3, Table 4, and Table 5. Table 3 presents the results obtained for a sample size $n = 2500$, Table 4 presents results for $n = 6600$, and Table 5 shows results for $n = 15200$. The last table differs from the previous ones by showing results for two different kinds of scrambling, i.e. *restricted form of Owen scrambling* and *combined scrambling.* The idea was to compare results for two different types of scrambling for relatively high value of n.

One can notice that the *combined* scrambling approach leads to smaller relative errors in comparison of the other algorithms for a number of points larger than 10000. That is why this algorithm has been chosen to be pre-

Table 3: 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 ($n = 2500, c \approx 0.51365, \delta \approx 0.08$).

Estimated quantity	Sobol quasi MC algorithm	Scrambled Sobol sequences	MCA-MSS	
			ρ	Rel. error
g_0	0.0002	0.0002	0.0007	0.0003
			0.002	0.0003
$\mathbf{D}$	0.0006	0.0076	0.0007	0.0017
			0.002	0.0044
S_1	0.0009	0.0041	0.0007	0.0034
			0.002	0.0006
S_2	0.0004	0.0088	0.0007	0.0039
			0.002	0.0008
S_3	0.2005	0.0047	0.0007	0.0566
			0.002	0.0851
S_1^{tot}	0.0012	0.0074	0.0007	0.0037
			0.002	0.0007
S_2^{tot}	0.0006	0.0055	0.0007	0.0041
			0.002	0.0009
S_3^{tot}	0.1152	0.1658	0.0007	0.0251
			0.002	0.0865
Time (s)	0.003	0.004	0.042	

Table 4: 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 ($n = 6600, c \approx 0.51365, \delta \approx 0.08$).

Estimated quantity	Sobol quasi MC algorithm	Scrambled Sobol sequences	MCA-MSS	
			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	0.0045	4**e**-05	0.0007	0.0001
			0.007	0.0031
S_2	0.0041	0.0007	0.0007	0.0012
			0.007	0.0014
S_3	0.0647	0.0217	0.0007	0.0193
			0.007	0.1066
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
Time (s)	0.006	0.008	0.114	

sented in Table 5. In the case of the smooth integrand - the rates of relative errors achieved applying these three scrambling approaches are the same (similar) for comparatively small number of points In the case of the non-smooth integrand - there are more essential differences of the relative errors of the approaches. For the integrands chosen here - one can point to number of points when scrambling algorithm(s) do not outperform the algorithm with the original Sobol points. But the scrambled algorithm (in particular, Owen scrambling modification) and Monte Carlo algorithm based on quasi-random sequences are more stable according to relative errors for comparatively small number of points in computing global sensitivity indices (see Table 3).

In case of smooth integrands (see Table 2, Table 3, Table 4, and Table 5) smaller or almost equal estimated computational complexity (time) appear for the corresponding scrambling algorithm in comparison to Sobol QMC algorithm. The reason is that the computational time needed for computing

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 ($n = 15200, c \approx 0.51365, \delta \approx 0.08$).

Estimated quantity	Sobol quasi MC algorithm	Modified Owen scrambling	Combined scrambling	MCA-MSS	
				ρ	Rel. error
g_0	2e-05	7e-05	6e-05	0.0007	3e-05
				0.007	0.0002
$\mathbf{D}$	6e-05	0.0003	7e-05	0.0007	0.0013
				0.007	0.0099
S_1	0.0001	0.0006	0.0002	0.0007	0.0002
				0.007	0.0013
S_2	0.0003	0.0013	9e-05	0.0007	0.0003
				0.007	0.0013
S_3	0.0006	0.0517	0.0188	0.0007	0.0486
				0.007	0.0125
S_1^{tot}	0.0002	0.0008	6e-05	0.0007	0.0002
				0.007	0.0011
S_2^{tot}	9e-05	0.0009	0.0002	0.0007	0.0005
				0.007	0.0014
S_3^{tot}	0.0119	0.0776	0.0088	0.0007	0.0388
				0.007	0.0210
Time (s)	0.015	0.018	0.017	0.253	

of a value of the integrand in this case is much shorter than the computational time needed for points generation.

The results about relative error in computing second-order sensitivity indices applying MC algorithm using modified Sobol's points, modified Owen scrambling approach of Sobol's points, and quasi-Monte Carlo algorithm using $\Lambda\Pi_\tau$ points are presented on Figure 1. The reference values for these indices are respectively $S_{12} \approx 0.00537, S_{13} \approx 0.000096$, and $S_{23} \approx 0.00018$. The algorithm proposed in [23] has been applied to compute the approximate values of second-order sensitivity indices of inputs of UNI-DEM. The estimated quantities are small in values and a loss of accuracy appears following its definition. It explains different rate of relative value in computations of S_{12} and the other indices S_{13} and S_{23} using the same sample size. The results presented on Figure 1 demonstrate that MCA-MSS leads to smaller relative error for fixed moderate values of sample size (from $n = 5000$ to $n = 25000$).

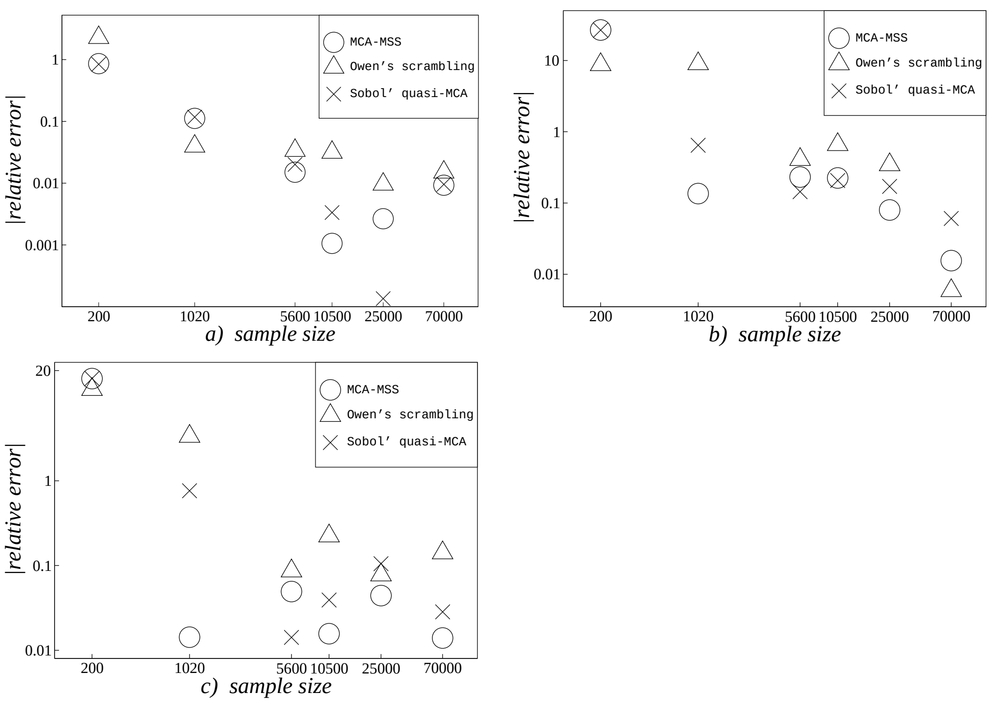

Figure 1: Relative errors (in absolute value) for numerical approximation of second-order sensitivity indices: a) S_{12}; b) S_{13}; c) S_{23}.

On the other hand, one can see that Owen approach gives worst results for most of the chosen sample sizes.

7. Discussion

One can clearly observe that the algorithm based on modified Sobol sequences improves the error estimates for non-smooth integrands when the radius ρ is smaller than the minimal distance between $\Lambda\Pi_\tau$ points δ. Strongly speaking this approach is applicable if ρ is much smaller than δ. The implementation of the algorithm shows that this requirement is not very strong. Even for relatively large radiuses ρ the results are good. 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.

It should be mentioned here that for relatively low number of points (< 1000) the algorithm based on modified Sobol sequences gives results with a high accuracy. The relative error is approximately equal to 0.0038 for $n = 100$. For the same sample size the Sobol algorithm gives more than 10 times higher error. For $n = 1000$ our algorithm gives relative error $0.0004 - 0.0016$ depending on the parameter κ while the Sobol algorithm gives 0.0114. This is an important fact because one has a possibility to estimate the value of the integral with a relatively high accuracy using a small number of random points.

If one deals with smooth functions, then the algorithm based on modified Sobol sequences is definitely better than the Monte Carlo algorithm based on SFMT generator, but it is not better than Sobol algorithm based on $\Lambda\Pi_\tau$ points. Actually the results are very close to each other. This result is not unexpectable since the Sobol algorithm is known to be very good for smooth functions (especially, for not very high dimensions).

8. Conclusion

A comprehensive theoretical and empirical study of the Monte Carlo algorithm based on modified Sobol sequences has been done. The algorithm combines properties of two of the best available approaches - Sobol quasi-Monte Carlo integration and a high quality SFMT pseudo-random number generator. It has been proven that the Monte Carlo algorithm based on Sobol sequences MCA-MSS has an optimal rate of convergence for functions with continuous and bounded first derivative in terms of probability and mean square error.

A comparison with several scrambling approaches as well as with the Sobol quasi-Monte Carlo algorithm and the algorithm using SFMT generator has been provided for numerical integration of non-smooth and 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 based on modified Sobol sequences gives similar rates of the relative error in comparison with scrambling approaches. But there are many cases when MCA-MSS has essential advantages. It holds especially for small in values sensitivity indices. The latter case is crucial to provide reliable sensitivity analysis.

Acknowledgment

The research reported in this paper is partly supported by the Bulgarian NSF Grants DTK 02/44/2009, DCVP 02/1, and DO 02-215/2008.

[1] I. Antonov, V. Saleev, An Economic Method of Computing LP_τ-sequences, USSR Comput. Math. Phy. 19, 1979, 252-256.

[2] P. Bradley, B. Fox, Algorithm 659: Implementing Sobol's Quasi Random Sequence Generator. ACM Trans. Math. Software 14(1), 1988, 88-100.

[3] H. Chi, P. Beerli, D. Evans, M. Mascagni, On the scrambled Sobol' sequences. V.S. Sunderam et al. (eds.), ICCS 2005, LNCS 3516, 2005, 775-782.

[4] I. T. Dimov, Monte Carlo Methods for Applied Scientists. World Scientific, London, Singapore (2008).

[5] I. T. Dimov, I. Farago, A. Havasi, Z. Zlatev, Operator Splitting and Commutativity Analysis in the Danish Eulerian Model, Math. Comp. Sim. 67, 2004, 217233.

[6] I. T. Dimov, R. Georgieva, Monte Carlo Method for Numerical Integration based on Sobol' Sequences, Springer, LNCS 6046, 2011, 50-59.

[7] I. T. Dimov, R. Georgieva, S. Ivanovska, Tz. Ostromsky, Z. Zlatev, Studying the Sensitivity of Pollutants' Concentrations Caused by Variations of Chemical Rates, Journal of Computational and Applied Mathematics 235, 2010, 391 - 402.

[8] I. Dimov, Tz. Ostromsky, Z. Zlatev, Challenges in using splitting techniques for large-scale environmental modeling, in: Advances in Air Pollution Modeling for Environmental Security (Farago, I., Georgiev, K., Havasi, A. - eds.) NATO Science Series 54, 2005, Springer, 115132.

[9] H. Faure, S. Tezuka, Another Random Scrambling of Digital (t,s)-sequences Monte Carlo and Quasi-Monte Carlo Methods, Springer-Verlag, Berlin, Germany (K. Fang, F. Hickernell, H. Niederreiter, eds.), 2000.

[10] S. Joe, F. Kuo, Constructing Sobol' Sequences with Better Two-dimensional Projections. SIAM J. Sci. Comput. 30, 2008, 2635-2654.

[11] T. Homma, A. Saltelli, Importance Measures in Global Sensitivity Analysis of Nonlinear Models, Reliability Engineering and System Safety 52, 1996, 1-17.

[12] H. Hong, F. Hickernell, Algorithm 823: Implementing Scrambled Digital Sequences ACM Trans. Math. Software 29:2, 2003, 95-109.

[13] P. L'Ecuyer, C. Lecot, B. Tuffin, A Randomized Quasi-Monte Carlo Simulation Method for Markov Chains. Operations Research 56(4), 2008, 958-975.

[14] P. L'Ecuyer, C. Lemieux, 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, 2002, 419-474, Kluwer Academic Publishers, Boston.

[15] Y. Levitan, N. Markovich, S. Rozin, I. Sobol', On Quasi-random Sequences for Numerical Computations, USSR Comput. Math. and Math. Phys. 28(5), 755-759 (1988).

[16] G. I. Marchuk, Mathematical Modeling for the Problem of the Environment, Studies in Mathematics and Applications, No. 16, North-Holland, Amsterdam, 1985.

[17] J. Matousek, On the L_2-discrepancy for Anchored Boxes. Journal of Complexity, 14: 527-556, 1998.

[18] H. Niederreiter, Low-Discrepancy and Low-Dispersion Sequences. Journal of Number Theory 30, 51-70 (1988).

[19] A. Owen, Randomly Permuted (t, m, s)-nets and (t, s)-sequences. Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, 106 in Lecture Notes in Statistics: 299-317, 1995.

[20] A. Owen, Scrambled Net Variance for Integrals of Smooth Functions. Ann. Statist., 25, 1541-1562, 1997.

[21] A. Owen, Variance and Discrepancy with Alternative Scramblings. ACM Trans. on Computational Logic., V: 1-16, 2002.

[22] M. Saito, M. Matsumoto, 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, 607-622. Springer (2008).

[23] A. Saltelli, Making best use of model evaluations to compute sensitivity indices, Computer Physics Communication 145, 2002, 280-297.

[24] A. Saltelli, K. Chan, M. Scott, Sensitivity Analysis, John Wiley and Sons publishers, Probability and Statistics series (2000).

[25] A. Saltelli, S. Tarantola, F. Campolongo, M. Ratto, Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models, Halsted Press, New York, 2004.

[26] I. Sobol, Multidimensional Quadrature Formulae and Haar functions (in Russian). Nauka, Moscow (1969).

[27] I. Sobol, Monte Carlo Numerical Methods (in Russian). Nauka, Moscow (1973).

[28] I. Sobol, On the Systematic Search in a Hypercube. SIAM J. Numerical Analysis 16, 790-793 (1979).

[29] I. Sobol, On Quadratic Formulas for Functions of Several Variables Satisfying a General Lipschitz Condition. USSR Comput. Math. and Math. Phys. 29(6), 936-941 (1989).

[30] I. Sobol, Quasi - Monte Carlo Methods. In: Sendov, Bl., Dimov, I.T. (eds.) International Youth Workshop on Monte Carlo Methods and Parallel Algorithms 1989, 75-81. World Scientific, Singapore (1990).

[31] I. M. Sobol, Sensitivity estimates for nonlinear mathematical models. Mathematical Modeling and Computational Experiment **1** (1993), 407-414.

[32] I. M. Sobol, Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Mathematics and Computers in Simulation, **55** (1-3) (2001) 271-280.

[33] I. Sobol, E. Myshetskaya, Monte Carlo estimators for small sensitivity indices, Monte Carlo Methods and Applications 13(5-6), 2007, 455-465.

[34] S. Tezuka, Uniform Random Numbers, Theory and Practice. Kluwer Academic Publishers, IBM Japan, 1995.

[35] H. Weyl, Ueber die Gleichverteilung von Zahlen mod Eins. Math. Ann. 77(3), 313-352 (1916).

[36] www.nag.co.uk/numeric/CL/CLdescription.asp

[37] www.math.sci.hiroshima-u.ac.jp/$\sim$ m-mat/MT/SFMT/index.html

[38] Z. Zlatev, Computer treatment of large air pollution models, KLUWER Academic Publishers, Dorsrecht-Boston-London, 1995.

[39] Z. Zlatev, I. T. Dimov, Computational and Numerical Challenges in Environmental Modelling, Elsevier, Amsterdam, 2006.

[40] Z. Zlatev, I. T. Dimov, K. Georgiev, Modeling the Long-range Transport of Air Pollutants, IEEE Computational Science and Engineering, **1** (3) (1994), 45-52.

[41] Z. Zlatev, I. T. Dimov, K. Georgiev, Three-dimensional version of the Danish Eulerian Model, Zeitschrift für Angewandte Mathematik und Mechanik, **76** (1996) S4, 473-476.