Efficient Monte Carlo algorithms with applications to sensitivity analysis

Ivan Dimov

Bulgarian Academy of Sciences (BAS)
e-mail: ivdimov@bas.bg

Academic Science Fund Grant between Bulgaria and Hungary
"Novel Performance-efficient Sensitivity Analysis of a Large Air Pollution Models by using Efficient Numerical Algorithms and High-performance Parallel Supercomputers"

Budapest,
November 28, 2019

- Problem setting
  - Concept of sensitivity analysis
  - Computational complexity of algorithms
  - $\Lambda\Pi_\tau$ Sobol Sequences
  - Randomized Quasi-Monte Carlo (RQMC)
- Monte Carlo algorithms based on modified Sobol sequences
- Numerical experiments
  - Case-study: Unified Danish Eulerian Model (UNI-DEM)
- Discussion of applicability and concluding remarks

- Consider the following problem of integration:

$$S(f) := I = \int_{U^d} f(x)dx, \quad \text{where}$$

$$U^d \equiv [0,1]^d, \quad x \equiv (x_1, \ldots, x_d) \in U^d \subset \mathbb{R}^d, \quad f \in C(U^d).$$

- Quadrature formula $A = \sum_{i=1}^{n} c_i f(x^{(i)})$.
- Randomized quadrature formula $A^R = \sum_{i=1}^{n} \sigma_i f(\xi^{(i)})$.
  - Assume for a given r.v. θ one can prove that $E\theta = I$.
  - Monte Carlo approximation to the solution: $\bar{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} \theta^{(i)} \approx I$.

Definition

If I is the exact solution of the problem, then the probability error is the least possible real number R_n, for which $P = Pr\left\{|\bar{\theta}_n - I| \leq R_n\right\}$, where $0 < P < 1$.

Definition (*sensitivity analysis*, SA).

The study of how uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input.

A. Saltelli et al. *Global Sensitivity. The Primer*. John Wiley & Sons, Ltd (2008).

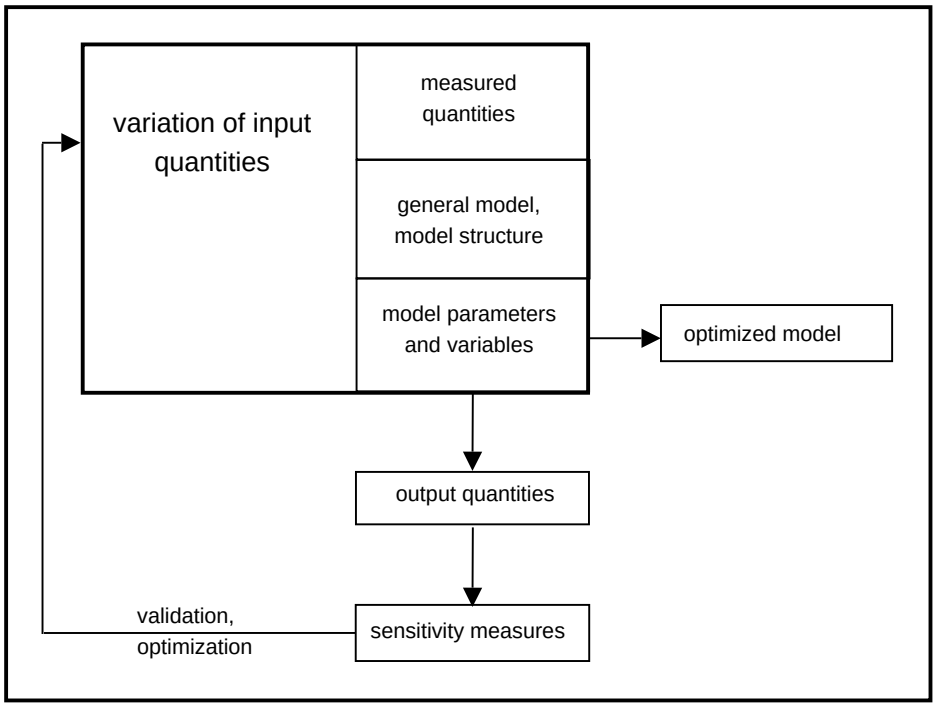

Figure: General procedure for sensitivity analysis.

- The mathematical model representation

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

is a vector of inputs with a joint p.d.f. $p(\mathbf{x}) = p(x_1, \ldots, x_d)$.

- The mathematical model representation

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

is a vector of inputs with a joint p.d.f. $p(\mathbf{x}) = p(x_1, \ldots, x_d)$.

- Total Sensitivity Index of input parameter x_i, $i \in \{1, \ldots, d\}$:

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

where

S_i - *the main effect (first-order sensitivity index)* of x_i and
$S_{il_1 \ldots l_{j-1}}$ – j^{th} *order sensitivity index for parameter* x_i $(2 \leq j \leq d)$.

- Approaches for Sensitivity analysis
  - Local approach (one-at-a-time experiments)
  - Screening methods
  - Variance-based methods - Sobol approach, FAST
  - Derivative-based global sensitivity measures

Table: Comparison of variance-based methods for evaluating global sensitivity indices.

Method	Cost (model runs)	Sensitivity measures
FAST (1973)	$O(d^2)$	$S_i, \forall i$
Sobol (1993)	$N(2d+2)$	$S_i, S_{x_i}^{tot}, \forall i$
EFAST (1999)	dN	$S_i, S_{x_i}^{tot}, \forall i$
Saltelli (2002)	$N(d+2)$	$S_i, S_{x_i}^{tot}, \forall i, \mathbf{D}^c_{-lj}, \forall l, j, l \neq j$
Saltelli (2002)	$N(2d+2)$	$S_i, S_{x_i}^{tot}, \forall i, \mathbf{D}^c_{lj}, \mathbf{D}^c_{-lj}, \forall l, j, l \neq j$

$$\mathbf{D}^c_{l_1} = \mathbf{D}_{l_1}, \quad \mathbf{D}^c_{l_1 l_2} = \mathbf{D}_{l_1} + \mathbf{D}_{l_2} + \mathbf{D}_{l_1 l_2},$$

$$\mathbf{D}_{-l_1 l_2} = \mathbf{D}_{j_1 j_2 \ldots j_{d-2}}, \text{ where } l_p \neq j_q \text{ for all } p \in [1,2], q \in [1,2,\ldots,d-2].$$

Variance-based Methods: Sobol Approach (1990)

ANalysis Of VAriances (ANOVA) HDMR of a square integrable function $f(\mathbf{x})$:

$$f(\mathbf{x}) = f_0 + \sum_{\nu=1}^{d} \sum_{l_1<\ldots<l_\nu} f_{l_1\ldots l_\nu}(x_{l_1}, x_{l_2}, \ldots, x_{l_\nu}), \quad \text{where } f_0 = const,$$

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

The functions in the right-hand side are defined in a unique way:

- $f_0 = \int_{U^d} f(\mathbf{x})\mathrm{d}\mathbf{x}, \quad f_{l_1}(x_{l_1}) = \int_{U^{d-1}} f(\mathbf{x}) \prod_{k\neq l_1} \mathrm{d}\mathbf{x}_k - f_0, \; l_1 \in \{1, \ldots, d\}$
- $\int_{U^d} f_{i_1\ldots i_\mu} f_{j_1\ldots j_\nu} \, \mathrm{d}\mathbf{x} = 0, \quad (i_1, \ldots, i_\mu) \neq (j_1, \ldots, j_\nu), \quad \mu, \nu \in \{1, \ldots, d\}.$

Definition (*global Sobol sensitivity indices*).

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

where

- partial variances $\mathbf{D}_{l_1 \ \ldots \ l_\nu} = \int f^2_{l_1 \ \ldots \ l_\nu} \mathrm{d}x_{l_1} \ldots \mathrm{d}x_{l_\nu},$
- total variance $\mathbf{D} = \int_{U^d} f^2(\mathbf{x}) \mathrm{d}\mathbf{x} - f_0^2, \qquad \mathbf{D} = \sum_{\nu=1}^{d} \sum_{l_1 < \ldots < l_\nu} \mathbf{D}_{l_1 \ldots l_\nu},$

and the following properties hold:

- $S_{l_1 \ \ldots \ l_s} \geq 0, \qquad \sum_{s=1}^{d} \sum_{l_1 < \ldots < l_s}^{d} S_{l_1 \ \ldots \ l_s} = 1.$

Approaches for Evaluating Small Sensitivity Indices

Let $\mathbf{x} = (\mathbf{y}, \mathbf{z}) \in \mathbb{R}^d, \mathbf{y} = (x_{k_1}, \ldots, x_{k_m}) \in \mathbb{R}^m, K = (k_1, \ldots, k_m)$.
Variance of the subset $\mathbf{y}$ (Sobol) : $\mathbf{D_y} = \sum_{n=1}^{m} \sum_{(i_1 < \ldots < i_n) \in K} \mathbf{D}_{i_1, \ldots, i_n}$.

- $S_y^{tot} = 1 - S_z$,
- Loss of accuracy if $\mathbf{D_y} << f_0^2$,
- Choose a constant $c \sim f_0$ and set the function $\varphi(\mathbf{x}) = f(\mathbf{x}) - c$,
- $\omega = \int \varphi(\mathbf{x}) \mathrm{d}\mathbf{x}$.

Approaches for Evaluating Small Sensitivity Indices

Initial Sobol Approach (I.M. Sobol, 1990)

$$\mathbf{D_y} = \int f(\mathbf{x})\, f(\mathbf{y}, \mathbf{z}')\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}' - f_0^2, \quad \mathbf{D} = \int_{U^d} f^2(\mathbf{x})\mathrm{d}\mathbf{x} - f_0^2.$$

Reducing the Mean Value (I.M. Sobol, 1990)

$$\mathbf{D_y} = \int \varphi(\mathbf{x})\, \varphi(\mathbf{y}, \mathbf{z}')\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}' - \omega^2, \quad \mathbf{D} = \int \varphi^2(\mathbf{x})\mathrm{d}\mathbf{x} - \omega^2.$$

Correlated Sampling (A. Saltelli, 2002)

$$\mathbf{D_y} = \int f(\mathbf{x})\, [f(\mathbf{y}, \mathbf{z}') - f(\mathbf{x}')]\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}', \quad \mathbf{D} = \int f(\mathbf{x})[f(\mathbf{x}) - f(\mathbf{x}')]\, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}'.$$

Combined Approach (A. Saltelli, 2002, S. Kucherenko, 2007)

$$\mathbf{D_y} = \int \varphi(\mathbf{x})[\varphi(\mathbf{y}, \mathbf{z}')\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}' - \varphi(\mathbf{x}')]\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}', \mathbf{D} = \int \varphi(\mathbf{x})[\varphi(\mathbf{x}) - \varphi(\mathbf{x}')]\, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}'.$$

- Consider the following integration problem:

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

where $\mathbf{x} \equiv (x_1, \ldots, x_d) \in U^d \subset \mathbb{R}^d$ and $f \in C(U^d)$ is an integrable function on U^d.

- Deterministic algorithms and Randomized (Monte Carlo) algorithms
- *The quadrature formula* $A^D(f,n) = \sum_{i=1}^{n} c_i f(\mathbf{x}^{(i)})$, defines an algorithm $A^D(f,n)$ that approximates the integral $S(f)$ with an integration error $err(f, A^D) \equiv \int_{U^d} f(\mathbf{x}) d\mathbf{x} - A^D(f,n)$
- *The randomized quadrature formula* $A^R(f,n) = \sum_{i=1}^{n} \sigma_i f(\xi^{(i)})$, defines an algorithm $A^R(f,n)$ that belongs to the class of randomized (Monte Carlo) denoted by $\mathcal{A}^{\mathcal{R}}$.

Definition.

Let d and k be integers, $d, k \geq 1$. We consider the class $F_0 \equiv \mathbf{W}^k(\|f\|; U^d)$ (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 $\dfrac{\partial^r f(\mathbf{x})}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}$, $\alpha_1 + \cdots + \alpha_d = r \leq k$, which are continuous when $r < k$ and bounded in sup norm when $r = k$. The semi-norm $\|\cdot\|$ on $\mathbf{W}^k$ is defined as

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

Definition.

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

Definition.

Consider the set $\mathcal{A}$ of algorithms A:

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

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

Uniformly distributed sequences

Definition (▸ H. Weyl, 1916).

The sequence $x_1, x_2, \ldots$ is called an uniformly distributed sequence (u.d.s.) if, for an arbitrary region $\Omega \subset U^s$,

$$\lim_{n\to\infty} [S_n(\Omega)/n] = V(\Omega),$$

where $S_n(\Omega)$ is the number of points with $1 \leq i \leq n$ that lie inside Ω and $V(\Omega)$ is the s-dimensional volume of Ω.

Theorem (▸ H. Weyl, 1916).

The relation

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(\xi_j) = \int_{\mathbf{E}^s} f(x)dx$$

holds for all Riemann integrable functions f if and only if the sequence $x_1, x_2, \ldots$ is u.d.s.

Uniformly distributed sequences

Definition (▸ H. Weyl, 1916).

The sequence $x_1, x_2, \ldots$ is called an uniformly distributed sequence (u.d.s.) if, for an arbitrary region $\Omega \subset U^s$,

$$\lim_{n\to\infty} [S_n(\Omega)/n] = V(\Omega),$$

where $S_n(\Omega)$ is the number of points with $1 \leq i \leq n$ that lie inside Ω and $V(\Omega)$ is the s-dimensional volume of Ω.

A u.d.s. should satisfy three additional requirements:

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

(ii) well distributed points for small n;

(iii) a computationally inexpensive algorithm.

(t, m, s)-nets and (t, s)-sequences in base b (▸ H. Niederreiter, 1988)

- The sum to converge towards the integral, the points x_i should fill U^s minimizing the holes. Another good property would be that the projections of x_i on a lower-dimensional face of U^s leave very few holes as well.

- An elementary s-interval in base b is 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 $j \in \{1, ..., s\}$.

(t, m, s)-nets and (t, s)-sequences in base b (▸ H. Niederreiter, 1988)

Definition (▸ H. Niederreiter, 1988).

Given two integers $0 \leq t \leq m$, a (t, m, s)-net in base b is a sequence x_n of b^m points of U^s such that $Card\ P \cap \{x_1, \ldots, x_{b^m}\} = b^t$ for any elementary interval P in base b of hypervolume $\lambda(P) = b^{t-m}$.

(t, m, s)-nets and (t, s)-sequences in base b (H. Niederreiter, 1988)

Definition (H. Niederreiter, 1988).

Given two integers $0 \leq t \leq m$, a (t, m, s)-net in base b is a sequence x_n of b^m points of U^s such that $Card\ P \cap \{x_1, \ldots, x_{b^m}\} = b^t$ for any elementary interval P in base b of hypervolume $\lambda(P) = b^{t-m}$.

Definition (H. Niederreiter, 1988).

Given a non-negative integer t, a (t, s)-sequence in base b is an infinite sequence of points x_n 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.

- **$\Lambda\Pi_\tau$ sequences**
  - Choose a primitive polynomial of degree s_j over the Galois field

$$P_j = x^{s_j} + a_{1,j}x^{s_j-1} + a_{2,j}x^{s_j-2} + \ldots + a_{s_j-1,j}x + 1,$$

    where the coefficients $a_{1,j}, \ldots, a_{s_j-1,j} \in \{0, 1\}$.
  - A sequence of positive integers $\{m_{1,j}, m_{2,j}, \ldots\}$ is defined by

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

    where the initial values $m_{k,j}, 1 \leq k \leq s_j$ are odd and less than 2^k.
  - The direction numbers $\{v_{1,j}, v_{2,j}, \ldots\}$: $v_{k,j} = \dfrac{m_{k,j}}{2^k}$.
  - The j-th component of the i-th point in a Sobol' sequence

$$x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \ldots,$$

    where i_k is the k-th binary digit of $i = (\ldots i_3 i_2 i_1)_2$.

▸ *I. Sobol' (1979), P. Bradley, B. Fox (1988)*

- Randomized Quasi-Monte Carlo turns QMC into a variance reduction method by carefully randomizing well distributed points $x_i \equiv (x_{i,1}, x_{i,2} \ldots x_{i,s})$.
- Examples of RQMC point sets include
  - randomly shifted lattice rules,
  - scrambled digital nets,
  - digital nets with a random digital shift,
  - a Latin hypercube sample or a stratified sample followed by a random permutation of the points.
- Array-RQMC.
  - *P. L'Ecuyer, C. Lecot, B. Tuffin (2008)*

Owen Nested Scrambling Algorithm

- 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. Let each $x_n^{(j)}$ is represented in base b as $x_n^{(j)} = (0.x_{n1}^{(j)}\ x_{n2}^{(j)} \ldots x_{nK}^{(j)} \ldots)_b$, K is the number of digits to be scrambled. Then nested scrambling proposed by Owen (▸ Owen, 1995, ▸ Owen, 2002) 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 $i \geq 2$.
- Computational complexity of implementation - nested scrambling requires b^{i-1} permutations to scramble the i-th digit.
- Various versions of scrambling methods based on the definitions of the π_i's: *Owen nested scrambling* (▸ Owen, 1995, ▸ Owen, 2002), *Tezuka's generalized Faure sequences* (▸ Tezuka, 1995), and *Matousek's linear scrambling* (▸ Matousek, 1998).
- 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 (▸ Owen, 1997).

- If

$$x_i = (x_{i,1}, x_{i,2} \ldots x_{i,s}) \in \mathbf{E}_i^s$$

- the i-th $\Lambda\Pi_\tau$ point,

then

the i-th random point $\xi_i(\rho)$ with a p.d.f. $p(x)$:

$$\xi_i(\rho) = x_i + \rho\omega_i,$$

where ω_i is a u.u.d vector in U^s and ρ is the "shaking radius".

- If

$$x_i = (x_{i,1}, x_{i,2} \ldots x_{i,s}) \in \mathbf{E}_i^s$$

- the i-th $\Lambda\Pi_\tau$ point,

then

the i-th random point $\xi_i(\rho)$ with a p.d.f. $p(x)$:

$$\xi_i(\rho) = x_i + \rho\omega_i,$$

where ω_i is a u.u.d vector in U^s and ρ is the "shaking radius".

- Example:

$$\omega_i = \{\cos\phi_i, \sin\phi_i\} \in \mathbf{E}_i^2$$

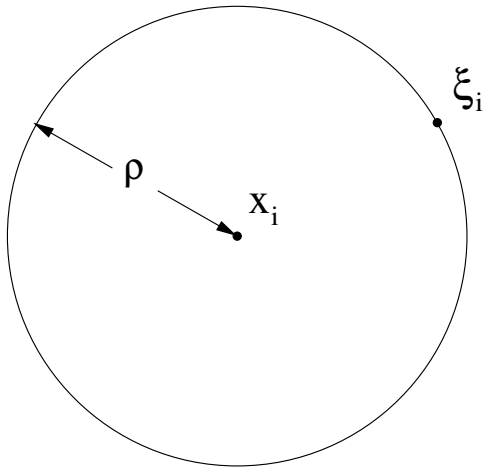

Figure: Generation of a random point $\xi_i \in \mathbf{E}_i^2$.

Theorem.

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

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

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

Remark.

One can see that the Monte Carlo algorithm MCA-MSS-1 has an optimal rate of convergence for functions with continuous and bounded first derivative [1].

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 $\mathcal{A}^{\mathcal{R}}$.

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

- The first pseudorandom point ξ_i is selected during the procedure used in MCA-MSS-1.
- The second pseudorandom point $\xi_i^{'}$ is chosen to be *symmetric* to ξ_i according to the central point s_i in each elementary subinterval $\mathbf{E}_i$.
- The value of the integral can be approximated in the following way:

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

- The first pseudorandom point ξ_i is selected during the procedure used in MCA-MSS-1.
- The second pseudorandom point $\xi_i^{'}$ is chosen to be *symmetric* to ξ_i according to the central point s_i in each elementary subinterval $\mathbf{E}_i$.
- The value of the integral can be approximated in the following way:

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

- Example:

$$\omega_i = \{\cos\phi_i, \sin\phi_i\} \in \mathbf{E}_i^2$$

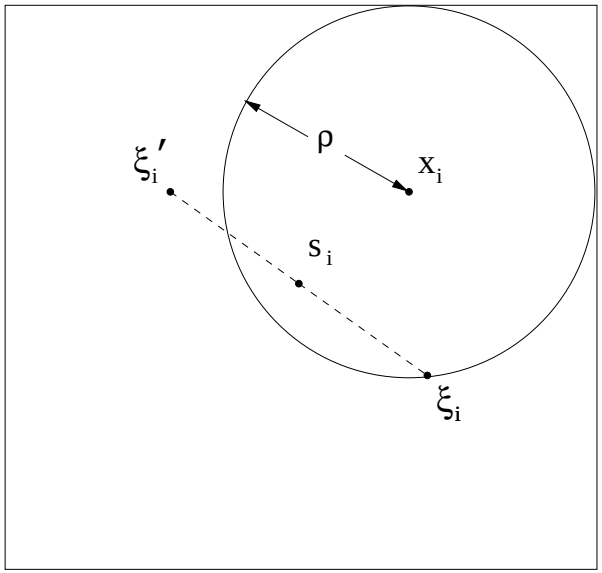

Figure: Generation of a random point $\xi_i^{'} \in \mathbf{E}_i^2$.

Theorem.

The quadrature formula $I(f) \approx \frac{1}{2m^d} \sum_{i=1}^{2n} \left[f(\xi_i) + f(\xi_i^{'}) \right]$ constructed for integrands f from $\mathbf{W}^2(L; U^d)$ satisfies

$$err(f,d) \leq \widetilde{c}_d^{\,\prime} \, \|f\| \, n^{-\frac{1}{2}-\frac{2}{d}} \quad \text{and} \quad r(f,d) \leq \widetilde{c}_d^{\,\prime\prime} \, \|f\| \, n^{-\frac{1}{2}-\frac{2}{d}},$$

where the constants $\widetilde{c}_d^{\,\prime}$ and $\widetilde{c}_d^{\,\prime\prime}$ do not depend on n.

Remark.

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. This means that the rate of convergence ($n^{-\frac{1}{2}-\frac{2}{d}}$) can not be improved for the functional class $\mathbf{W}^2$ in the class of the randomized algorithms $\mathcal{A}^{\mathcal{R}}$.

Sketch of proof. (1)

- One can use the d-dimensional Taylor formula to present the function $f(\mathbf{x}^{(j)})$ in E_j around the central point $\mathbf{s}^{(j)}$:

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

where $\nabla f(\mathbf{x}) = \left[\dfrac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \dfrac{\partial f(\mathbf{x})}{\partial x_d}\right]$ and $[D^2 f(\mathbf{x})] = \left[\dfrac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_k}\right]_{i,k=1}^{d}$.

- Thus, we get the following estimate of the variance

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

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

- Since $f \in \mathbf{W}^2(L; U^d)$:

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

Sketch of proof. (2)

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

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

where the following assumption holds for the probability density function

$$\int_{\mathbf{E}_j} p(\mathbf{x}) d\mathbf{x} = p_j \leq \frac{c_1^{(j)}}{n},$$

where $c_1^{(j)}$ are constants.

- The application of the Tchebychev's inequality to the variance yields

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

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

- The first pseudorandom point ξ_i is generated uniformly distributed inside $\mathbf{E}_i$.
- The second point $\xi_i^{'}$ is chosen to be *symmetric* to ξ_i according to the central point s_i in each elementary subinterval $\mathbf{E}_i$.
- The value of the integral can be approximated in the following way:

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

- The first pseudorandom point ξ_i is generated uniformly distributed inside $\mathbf{E}_i$.
- The second point $\xi_i^{'}$ is chosen to be *symmetric* to ξ_i according to the central point s_i in each elementary subinterval $\mathbf{E}_i$.
- The value of the integral can be approximated in the following way:

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

- Example:

$$\omega_i = \{\cos\phi_i, \sin\phi_i\} \in \mathbf{E}_i^2$$

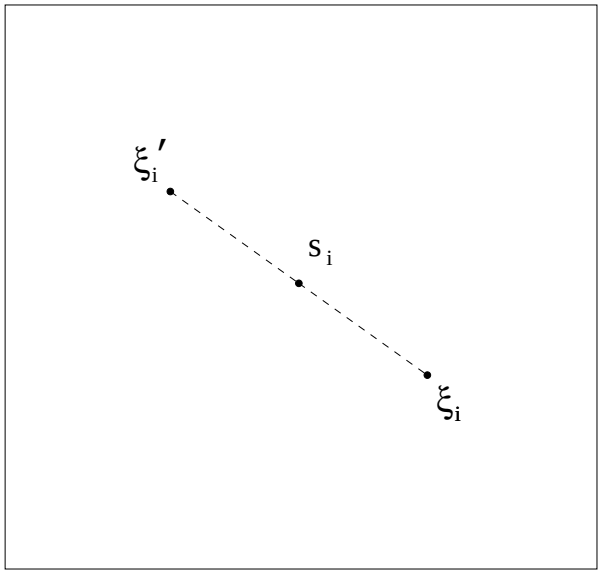

Figure: Generation of a random point $\xi_i^{'} \in \mathbf{E}_i^2$.

Example of
a non-smooth integrand:

$$f_1(x_1, x_2, x_3, x_4) = \sum_{i=1}^{4} |(x_i - 0.5)^{-1/3}|,$$

$$S(f_1) \approx 7.55953.$$

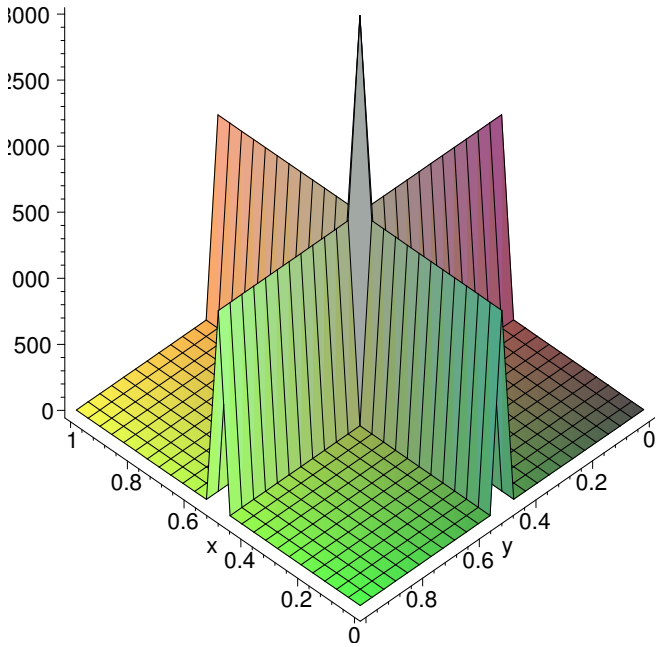

Figure: The integrand function in two-dimensional case.

Table: Radius ρ of spheres of the random points (*radius coefficient* $\kappa = \rho/\delta$)

n	Min. dist., δ	κ	ρ	κ	ρ	κ	ρ
10	0.43301	0.001	0.00043	0.09	0.03897	0.4	0.17321
10^2	0.13166	0.001	0.00013	0.09	0.01185	0.4	0.05266
10^3	0.06392	0.001	0.00006	0.09	0.00575	0.4	0.02557
10^4	0.02812	0.001	0.00003	0.09	0.00253	0.4	0.01125
50.10^3	0.01400	0.001	0.00001	0.09	0.00126	0.4	0.00560

Table: Relative error and computational time for numerical integration

n	▸ SFMT		▸ Sobol'		MCA				
	Rel. err.	Time (s)	Rel. err.	Time (s)	δ	κ	$\rho \times 10^3$	Rel. err.	Time (s)
10	0.0001	< 0.01	0.2813	< 0.01	0.433	0.03	13	0.0438	< 0.01
						0.45	195	0.0509	< 0.01
10^2	0.0114	0.01	0.0565	< 0.01	0.132	0.03	3.9	0.0038	0.01
						0.45	59	0.0050	0.01
10^3	0.0023	0.06	0.0114	0.01	0.064	0.03	1.9	0.0016	0.10
						0.45	29	0.0004	0.11
10^4	0.0006	0.53	0.0023	0.06	0.028	0.03	0.8	4**e**-05	3.56
						0.45	12.7	0.0002	3.58
30.10^3	0.0002	1.63	0.0011	0.19	0.019	0.03	0.6	0.0002	28.5
						0.45	8.3	0.0003	28.8
50.10^3	0.0009	2.67	0.0008	0.29	0.014	0.03	0.4	0.0002	74.8
						0.45	6.3	2**e**-05	75.7

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.

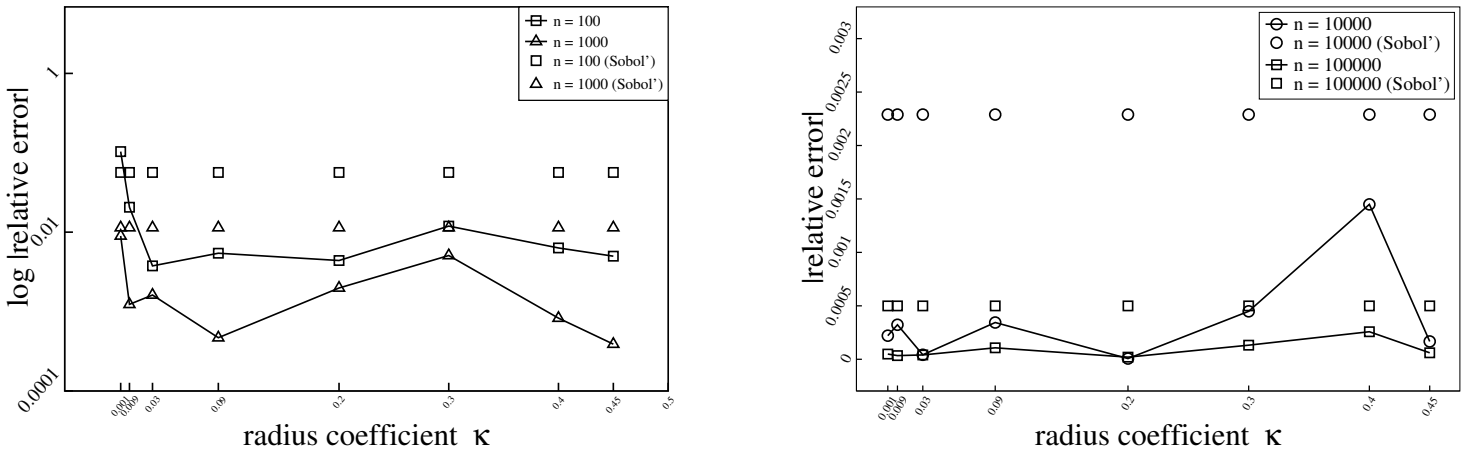

Figure: Relative error according to the "shaking radius".

Table: Difference of relative errors for Sobol' algorithm and the proposed Monte Carlo algorithm

n \ κ	0.009	0.03	0.2	0.45
10	0.07709	0.23746	0.20639	0.23037
10^2	0.03594	0.05277	0.05214	0.05155
10^3	0.01014	0.00976	0.00940	0.01099
10^4	0.00197	0.00225	0.00228	0.00212
30.10^3	0.00102	0.00094	0.00084	0.00079
50.10^3	0.00077	0.00062	0.00077	0.00078

Example of
a smooth integrand:

$$f_2(x_1, x_2, x_3, x_4) = \mathbf{e}^{x_1+2x_2} \frac{cos(x_3)}{1 + x_2 + x_3 + x_4},$$

$$S(f_2) \approx 1.83690.$$

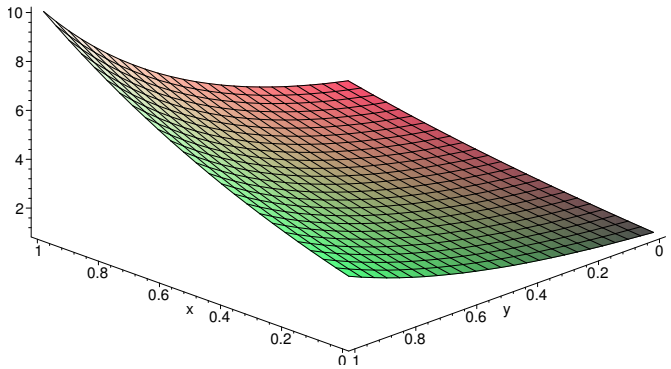

Figure: The integrand function in two-dimensional case.

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

n	**SFMT**		**Sobol QMCA**		**Owen scrambling**		**MCA-MSS-1**		
	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

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

# of points n	**Sobol QMCA**		**MCA-MSS-1**			**MCA-SMS-2**		**MCA-SMS-2-S**	
(# of double points $2n$)	Rel. error	Time (s)	ρ $\times 10^3$	Rel. error	Time (s)	Rel. error	Time (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	7e-06	1.340	9e-06	0.494
(2×2^{16})			1.2	0.0001	1.630	5e-06	1.380		

Table: Relative error and computational time for numerical integration

n	SFMT		Sobol'		MCA				
	Rel. err.	Time (s)	Rel. err.	Time (s)	δ	κ	$\rho \times 10^3$	Rel. err.	Time (s)
10^2	0.0350	< 0.01	0.0155	< 0.01	0.132	0.03	3.9	0.0160	0.01
						0.45	59	0.0264	0.01
10^3	0.0045	0.01	0.0023	< 0.01	0.064	0.03	1.9	0.0025	0.06
						0.45	29	0.0058	0.06
10^4	0.0016	0.10	0.0002	0.02	0.028	0.03	0.8	0.0003	3.29
						0.45	12.7	0.0016	3.28
30.10^3	0.0006	0.28	0.0001	0.04	0.019	0.03	0.6	0.0002	28.5
						0.45	8.3	0.0011	28.4
50.10^3	0.0004	0.46	6**e**-05	0.07	0.014	0.03	0.4	0.0001	76.0
						0.45	6.3	0.0008	76.1

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.

- **Example** of a non-smooth integrand:

$$f_1(x_1, x_2, x_3, x_4) = \sum_{i=1}^{4} |(x_i - 0.8)^{-1/3}|, \qquad S(f_1) \approx 7.22261.$$

- **Example** of a non-smooth integrand:

$$f_1(x_1, x_2, x_3, x_4) = \sum_{i=1}^{4} |(x_i - 0.8)^{-1/3}|, \qquad S(f_1) \approx 7.22261.$$

- **Example** of a smooth integrand:

$$f_2(x_1, x_2, x_3, x_4) = x_1 \; x_2^2 \; \mathbf{e}^{x_1 x_2} \sin x_3 \cos x_4, \qquad S(f_2) \approx 0.10897.$$

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

n	**Sobol QMCA**		**MCA-MSS-1**			**MCA-MSS-2-S**	
	Rel. error $\times 10^3$	Time (s)	ρ	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
(2592)			4.1	0.0046	0.030		
2×8^4	0.0004	0.004	0.9	0.0008	0.090	0.0002	0.025
(8192)			2.9	0.0024	0.090		
2×10^4	0.0002	0.008	0.6	0.0001	0.220	5e-05	0.070
(20000)			2.0	0.0013	0.230		
2×13^4	0.0001	0.022	0.4	0.0001	0.630	4e-06	0.178
(57122)			1.2	0.0007	0.640		
2×14^4	5e-06	0.029	0.4	1e-05	0.860	1e-05	0.237
(76832)			1.2	0.0005	0.880		
2×15^4	8e-06	0.036	0.4	0.0001	1.220	9e-07	0.313
(101250)			1.2	0.0005	1.250		

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

n	SFMT		Sobol QMCA		Owen scrambling		MCA-MSS-1		
	Rel. error	Time (s)	Rel. error	Time (s)	Rel. error	Time (s)	ρ $\times 10^3$	Rel. error	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

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s})c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} +$$
$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) +$$
$$+E_s + Q_s(c_1, c_2, \ldots, c_q) - (k_{1s} + k_{2s}) c_s, \quad s = 1, 2, \ldots, q.$$

q	–	number of equations = number of chemical species,
c_s	–	concentrations of the chemical species,
u, v, w	–	components of the wind along the coordinate axes,
K_x, K_y, K_z	–	diffusion coefficients,
E_s	–	emissions in the space domain,
k_{1s}, k_{2s}	–	coefficients of dry and wet deposition respectively,
$Q_s(c_1, c_2, \ldots, c_q)$	–	non-linear functions that describe the chemical reactions between species.

Table: 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 QMCA**	**Owen scrambling**	**MCA-MSS-1**	
			ρ	**Rel. error**
g_0	1e-05	0.0001	0.0007	0.0001
			0.007	6e-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	6e-05	0.0007	2e-05
			0.007	0.0034
s_3^{tot}	0.0259	0.0102	0.0007	0.0099
			0.007	0.0211

Example (integrand with computational irregularities).

$$f(x) = (1 + \sum_{i=1}^{d} a_i \, x_i)^{-(d+1)}$$

$$||a||_1 = \frac{600}{d^2}$$

$$I[f] = \int_{U^d} f(x) \mathrm{d}\mathbf{x}$$

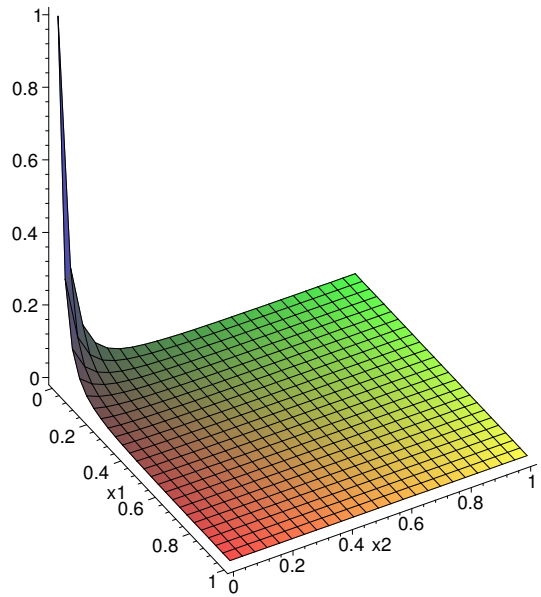

Figure: Genz integrand function with a corner peak in two dimensions.

Table: Relative error and CPU time for dimension $d = 5$, $I[f] = 0.21214\mathbf{e} - 05$, $a = (5, 5, 5, 5, 4)$.

Adaptive Monte Carlo Algorithm				Plain Monte Carlo Algorithm			
N	$I_N[f]$ $\times 10^5$	Rel. error	Time (s)	N	$I_N[f]$ $\times 10^5$	Rel. error	Time (s)
100	0.213	0.008	0.01	94.10^2	0.18	0.13	0.01
1000	0.211	0.007	0.13	94.10^3	0.19	0.08	0.06
10000	0.212	0.001	1.42	94.10^4	0.22	0.02	0.55
100000	0.212	0.0009	14.05	94.10^5	0.20	0.04	5.38

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.

Table: Relative error and CPU time for dimension $d = 18$, $I[f] = 0.99186\mathbf{e} - 05$, $a = \left(\frac{1}{9}, \frac{2}{27}, \frac{2}{27}, \frac{1}{9}, \frac{2}{27}, \frac{1}{9}, \frac{1}{9}, \frac{4}{27}, \frac{2}{27}, \frac{1}{9}, \frac{1}{9}, \frac{2}{27}, \frac{2}{27}, \frac{1}{9}, \frac{1}{9}, \frac{4}{27}, \frac{1}{9}, \frac{1}{9}\right)$.

Adaptive Monte Carlo Algorithm				Plain Monte Carlo Algorithm			
N	$I_N[f]$ $\times 10^5$	Rel. error	Time (s)	N	$I_N[f]$ $\times 10^5$	Rel. error	Time (s)
10	0.9923	0.0005	7	2621440	0.989	0.002	6
100	0.9918	0.00005	75	26214400	0.909	0.084	60
1000	0.9919	0.00008	758	262144000	0.510	0.48	600

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.

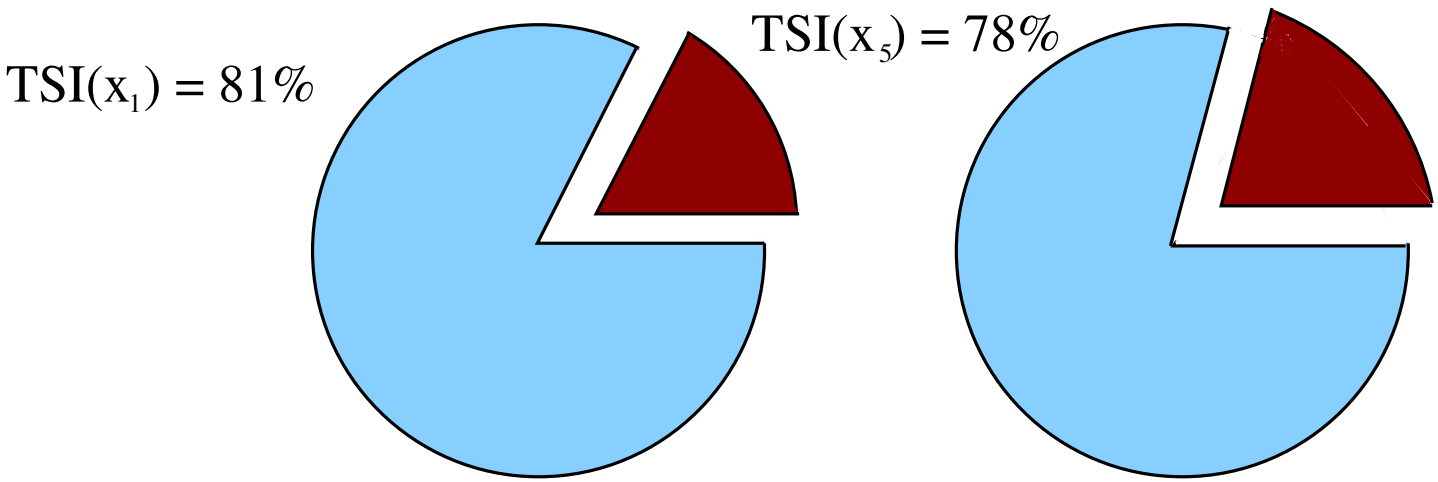

Figure: Total sensitivity indices of input parameters.

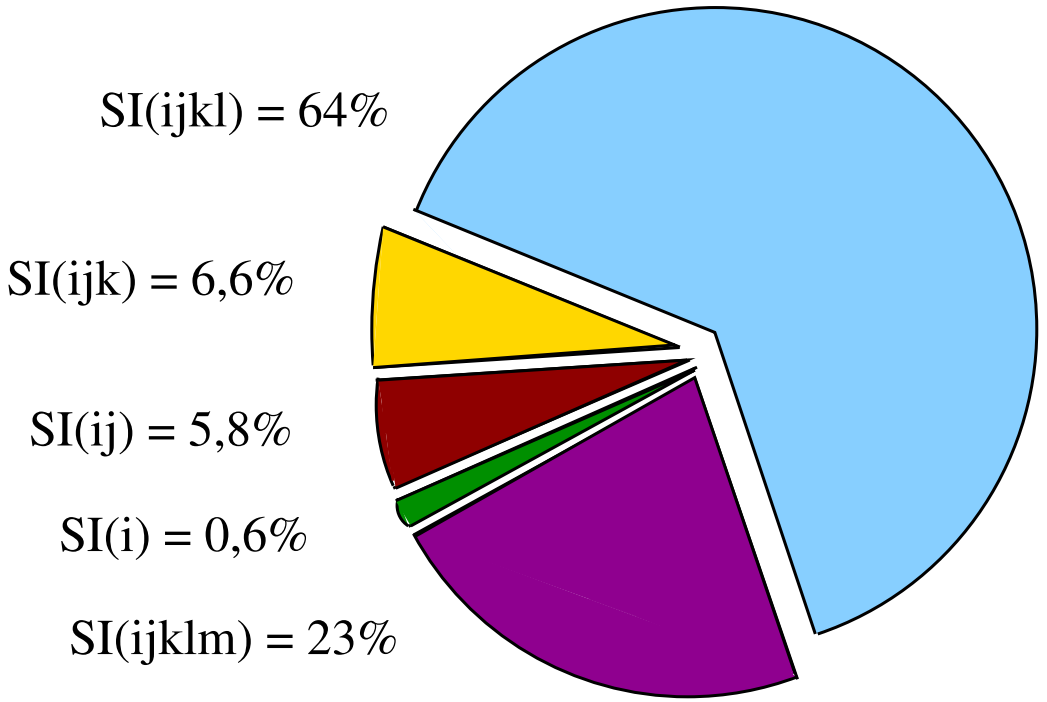

Figure: First-, second-, third-, fourth-, fifth-order effects.

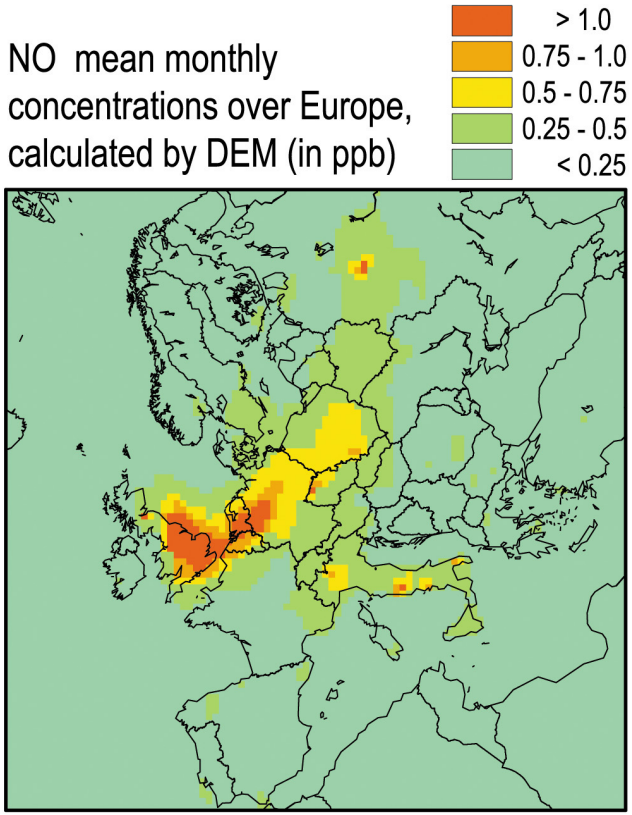

Figure: European nitrogen oxides concentrations

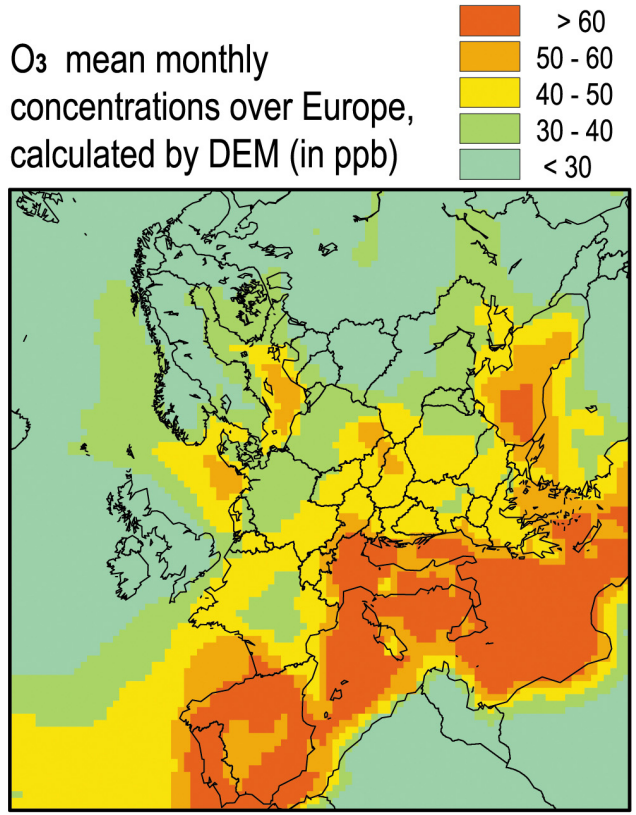

Figure: European ozone concentrations

Table: Total sensitivity indices of input parameters obtained using different variance-based approaches for sensitivity analysis.

approach / estimated quantity	Standard (Sobol')		Approaches for small indices	
			red. of the m.v.	combined
	$\mathbf{x} \in [0.1; 2.0]^3$	$\mathbf{x} \in [0.6; 1.4]^3$	$\mathbf{x} \in [0.6; 1.4]^3$	$\mathbf{x} \in [0.6; 1.4]^3$
integrand $g(\mathbf{x})$	$f(\mathbf{x})$	$f(\mathbf{x})$	$f(\mathbf{x}) - c$	$f(\mathbf{x}) - c$
c	-	-	0.51737	0.51737
g_0	0.51520	0.51634	0.25145	0.25145
$\mathbf{D}$	0.26181	0.26446	0.07061	0.00530
S_1	0.26386	0.26530	0.27354	0.52979
S_2	0.26447	0.26359	0.26713	0.46142
S_3	0.25348	0.25209	0.22406	0.00222
$\sum_{i=1}^{3} S_i$	0.78182	0.78097	0.76474	0.99342

Table: Total sensitivity indices of input parameters obtained using different variance-based approaches for sensitivity analysis.

approach / estimated quantity	Standard (Sobol')		Approaches for small indices	
			red. of the m.v.	combined
	$\mathbf{x} \in [0.1; 2.0]^3$	$\mathbf{x} \in [0.6; 1.4]^3$	$\mathbf{x} \in [0.6; 1.4]^3$	$\mathbf{x} \in [0.6; 1.4]^3$
S_{12}	0.06885	0.06941	0.07994	0.00628
S_{13}	0.06598	0.06634	0.06845	0.00009
S_{23}	0.06613	0.06592	0.06686	0.00021
$\sum_{i,j=1,i\leq j}^{3} S_{ij}$	0.20096	0.20167	0.21525	0.00658
S_{123}	0.01722	0.01736	0.02001	0.000003
$S^{tot}_{x_1}$	0.41592	0.41841	0.44195	0.53615
$S^{tot}_{x_2}$	0.41667	0.41627	0.43395	0.46791
$S^{tot}_{x_3}$	0.40281	0.40170	0.37938	0.00252

- *The proposed algorithms 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 the proposed approach is applicable if ρ is much smaller than δ. The implementation of the algorithms 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.

- For relatively low number of points (< 1000) the proposed algorithms 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 algorithms 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*.

- *The proposed algorithms 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 the proposed approach is applicable if ρ is much smaller than δ. The implementation of the algorithms 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.

- For relatively low number of points (< 1000) the proposed algorithms 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 algorithms 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*.

- The algorithms based on modified Sobol sequences combines properties of *two of the best available approaches* - Sobol quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.
- It has been proven that the Monte Carlo algorithm *MCA-MSS-1* has *an optimal rate of convergence for functions with continuous and bounded first derivatives in terms of probability and mean square error.*
- It has been proven that the Monte Carlo algorithm *MCA-MSS-2* has *an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error.*
- 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.

- The algorithms based on modified Sobol sequences combines properties of *two of the best available approaches* - Sobol quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.
- It has been proven that the Monte Carlo algorithm *MCA-MSS-1* has *an optimal rate of convergence for functions with continuous and bounded first derivatives in terms of probability and mean square error.*
- It has been proven that the Monte Carlo algorithm *MCA-MSS-2* has *an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error.*
- 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.

- The algorithms based on modified Sobol sequences combines properties of *two of the best available approaches* - Sobol quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.
- It has been proven that the Monte Carlo algorithm *MCA-MSS-1* has *an optimal rate of convergence for functions with continuous and bounded first derivatives in terms of probability and mean square error.*
- It has been proven that the Monte Carlo algorithm *MCA-MSS-2* has *an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error.*
- 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.

- The algorithms based on modified Sobol sequences combines properties of *two of the best available approaches* - Sobol quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.
- It has been proven that the Monte Carlo algorithm *MCA-MSS-1* has *an optimal rate of convergence for functions with continuous and bounded first derivatives in terms of probability and mean square error.*
- It has been proven that the Monte Carlo algorithm *MCA-MSS-2* has *an optimal rate of convergence for functions with continuous and bounded second derivatives in terms of probability and mean square error.*
- 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.

- Bradley, P., Fox, B.: Algorithm 659: Implementing Sobol's Quasi Random Sequence Generator. ACM Trans. Math. Software 14(1), 88–100 (1988)
- Joe, S., Kuo, F.Y.: Constructing Sobol Sequences with Better Two-dimensional Projections. SIAM J. Sci. Comput. 30, 2635–2654 (2008).
- L'Ecuyer, P., Lemieux, C.: Recent Advances in Randomized Quasi-Monte Carlo Methods. In: Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer Academic Publishers, Boston (2002)
- P. L'Ecuer, C. Lecot, B. Tuffin (2008): A Randomized Quasi-Monte Carlo Simulation Method for Markov Chains. Operations Research, 56, 4 (2008), 958-975.
- J. Matousek: On the L_2-discrepancy for Anchored Boxes. Journal of Complexity, 14: 527-556, 1998.
- Niederreiter, H.: Low-Discrepancy and Low-Dispersion Sequences. Journal of Number Theory 30, 51–70 (1988)
- 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.
- A. Owen: Scrambled Net Variance for Integrals of Smooth Functions. Ann. Statist., 25, 1541-1562, 1997.
- A. Owen: Variance and Discrepancy with Alternative Scramblings. ACM Trans. on Computational Logic., V: 1-16, 2002.
- Sobol', I.: On the Systematic Search in a Hypercube. SIAM J. Numerical Analysis 16, 790–793 (1979)
- Sobol', I.: Quasi - Monte Carlo Methods. In: Sendov, Bl., Dimov, I.T. (eds.) International Youth Workshop on Monte Carlo Methods and Parallel Algorithms 1989, pp. 75–81. World Scientific, Singapore (1990)
- S. Tezuka: Uniform Random Numbers, Theory and Practice. Kluwer Academic Publishers, IBM Japan, 1995.