Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Statistical Planning and Inference 136 (2006) 3231–3247

journal of statistical planning and inference

www.elsevier.com/locate/jspi

A study on algorithms for optimization of Latin hypercubes

M. Liefvendahl[a], R. Stocki[b],*

[a]Swedish Defence Research Agency, FOI, Grindsjön Research Center, 147 25 Tumba, Sweden
[b]Institute of Fundamental Technological Research, Polish Academy of Sciences, Swietokrzyska 21, 00-049 Warsaw, Poland

Received 24 January 2003; accepted 25 January 2005
Available online 18 April 2005

Abstract

A crucial component in the statistical simulation of a computationally expensive model is a good design of experiments. In this paper we compare the efficiency of the columnwise–pairwise (CP) and genetic algorithms for the optimization of Latin hypercubes (LH) for the purpose of sampling in statistical investigations. The performed experiments indicate, among other results, that CP methods are most efficient for small and medium size LH, while an adopted genetic algorithm performs better for large LH.

Two optimality criteria suggested in the literature are evaluated with respect to statistical properties and efficiency. The obtained results lead us to favor a criterion based on the physical analogy of minimization of forces between charged particles suggested in Audze and Eglais (1977. Problems Dyn. Strength 35, 104–107) over a ‘maximin distance’ criterion from Johnson et al. (1990. J. Statist. Plann. Inference 26, 131–148).
© 2005 Elsevier B.V. All rights reserved.

MSC: 62K05; 05B15

Keywords: Design of experiment; Latin hypercube; Genetic algorithm

* Corresponding author.
E-mail address: rstocki@ippt.gov.pl (R. Stocki).

doi:10.1016/j.jspi.2005.01.007

1. Introduction

In this article, concerning design of experiments (DOE), we take the point of view of computer simulation of stochastic phenomena. By this we mean that we have a (deterministic) computer model of a physical phenomenon and some model parameters are given as random variables (as opposed to fixed numbers). We thus want to first sample the random variables, then, for each sample point, run a computer simulation and finally analyze the stochastic properties of the solution.

Some of the algorithms evaluated in this paper have been implemented in the M-XPLORE module of the RADIOSS software (Braibant et al., 2002). This module is primarily aimed at stochastic simulations of car crash simulation. The results of this paper are, of course, of interest in a much wider spectrum of applications, the important feature being only that the simulation is computationally expensive so that only few (say in the range 20–500) samples are affordable.

This paper is focused on sampling using optimized Latin hypercubes (OLH). In Section 2, the definition of an Latin hypercubes (LH) and the optimality criteria for OLH are presented. It is well-known that OLH is a viable choice for design of experiments when one considers the following aspects.

Statistical optimality: There are many measures of statistical optimality of a DOE. Most of them are based on fitting a (stochastic) model to experiments/computed data. This point of view is taken in e.g. Mitchell (1974) and Park (1994) and leads to optimality criteria often referred to as entropy criteria. The important theoretical paper (Johnson et al., 1990) also takes this point of view, but in addition introduces 'minimax' and 'maximin' distance criteria. It is then proved that asymptotically the introduced criteria (which are simpler to compute than entropy criterion) give similar designs for large problems as the entropy-based optimal DOE. A second point of view concerning optimality criteria is to study how well the statistical properties of some model are predicted/computed. This is used in the present paper and also in the pioneering article (McKay et al., 1979) where LH were first introduced and (experimentally) shown to be competitive. In the current paper, a convenient optimality criterion introduced in Audze and Eglais (1977) as well as the maximin criterion are used to evaluate the quality of the resulting DOE.

Projection properties: By good projection properties we mean that the sample points should be well spread out when projected onto a subspace spanned by a number of coordinate axes. This is often desired in the applications when one does not know a priori if some random variables have a negligible effect on the response of the system.

Once generated, an OLH for p variables and N points is independent of the considered application. It is stored in a matrix and does not need to be computed again.

Although for small LH the computational cost of finding OLH is negligible compared to the expensive computer simulation of a physical phenomenon, it grows very fast with the number of samples and variables. For large LH (hundreds of samples and tens of variables) it may even take hours and days with fast computers. The computational cost depends, of course, on the algorithms used for OLH optimization and the adopted optimality criterion. In this paper, the CP algorithm invented in Park (1994) (with the modification described in Ye et al. (2000)) and a genetic algorithm, inspired by the algorithm proposed in Simpson

(1998), are compared. There are more methods for LH optimization in the literature, which seem competitive. In Ye et al. (2000), the CP algorithm was proposed for symmetric OLH and in Morris and Mitchell (1995), the simulated annealing algorithm was used for LH optimization.

There are two main purposes of the current paper. First, to compare optimality criteria for the optimization of LH with respect to the prediction accuracy of statistical moments of the output. This is done in the computational experiments of Section 4 where we, for two examples, evaluate the accuracy of the mean estimates, for different methods, criteria and sample sizes. The second main purpose is to compare the efficiency of the CP and the genetic algorithms for LH optimization. Some indications for the choice of optimality criterion and optimization algorithm will be based on the study of two numerical examples.

The fundamental problem that is addressed in this paper is to distribute points as uniformly as possible in a hypercube (taking into account statistical optimality and projection properties). This problem also arises for quasi-Monte Carlo methods for multi-dimensional integration. In this area the research has focused on so-called low-discrepancy sequences, see e.g. Niederreiter (1992) and Caflisch (1998) for an overview of this line of research. We perform comparative tests for OLH and Sobol sequences (which are of the low-discrepancy type) generated by the free software, which is described in Bratley and Fox (1988) and which is available from Netlib (`www.netlib.org`). These tests are presented in Sections 4.1 and 4.4.

2. Description of algorithms

An LH is given by an $N \times p$-matrix (i.e. a matrix with N rows and p columns) $\boldsymbol{L}$, where each column of $\boldsymbol{L}$ consists of a permutation of the integers 1 to N. We will refer to each row of $\boldsymbol{L}$ as a (discrete) sample point and use the notation

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \vdots \\ x_{N1} & \cdots & x_{Np} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{x}_j$ is the jth sample point.

Our ultimate goal is to obtain N points in the parameter space $\mathbb{R}^p$. The sample must, of course, be generated taking into account the probability distributions of the parameters. The most difficult task, however, turns out to be the construction of an optimized LH. In Section 2.2, the CP method for LH optimization is presented and in Section 2.3, the genetic algorithm is described. Before these, in Section 2.1 we state a number of optimality criteria and formulate the optimization problem. Section 2.4 describes the convergence criteria used in the computations and in Section 2.5 it is shown how to use LH for generating samples of continuous random variables.

For the generation of Sobol sequences, which are based on completely different ideas, we refer to Bratley and Fox (1988) and the references therein. In Section 2.6 we collect some remarks on the algorithm we use for the tests.

2.1. Optimality criteria

We will now introduce three functions from the set of LH to $\mathbb{R}$. These functions will subsequently be used in the criteria with respect to which LH will be optimized.

The first function d is defined to be the square of the minimum (Euclidean) distance between sample points

$$d(\boldsymbol{L}) := \min_{1 \leqslant i,j \leqslant N, i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2. \tag{2}$$

We define another function n to be the number of occurrences of the minimum distance. For a given LH, this number is thus denoted by $n(\boldsymbol{L})$.

Next we introduce a function G, which, in a physical analogy, is the sum of the norm of the repulsive forces if the sample points are considered as electrically charged particles (see Audze and Eglais, 1977; Bates et al., 2003).

$$G(\boldsymbol{L}) := \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{\underbrace{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}}. \qquad \text{Electric Force} \tag{3}$$

From the point of view of the physical analogy, it would have been natural with power 1 (instead of 2) in the denominator of the terms of the sum. However, with power 2, a computation of a square root for each term is avoided. This has a noticeable effect on the execution speed since the function G will be evaluated many times in the inner loop during optimization.

The above three functions will now be used to formulate the criteria we use for LH optimization.

Minimum distance: According to this criterion, an LH $\boldsymbol{L}_1$ is considered better than an LH $\boldsymbol{L}_2$ if $d(\boldsymbol{L}_1) > d(\boldsymbol{L}_2)$. If $d(\boldsymbol{L}_1) = d(\boldsymbol{L}_2)$, then $\boldsymbol{L}_1$ is better than $\boldsymbol{L}_2$ if $n(\boldsymbol{L}_1) < n(\boldsymbol{L}_2)$.

Force: $\boldsymbol{L}_1$ is better than $\boldsymbol{L}_2$ if $G(\boldsymbol{L}_1) < G(\boldsymbol{L}_2)$.

We remark that, for the minimum distance criterion, it is possible to check for equality even in floating point arithmetic. This is so, because one can compare the square of the minimum distances, which is an integer (since all points have integer coordinates).

2.2. The columnwise–pairwise algorithm

A first-order modification of a column of an LH matrix is defined as an interchange of two of its elements.

Now, in pseudo-code, we describe the idea of the so-called ‘CP-sweep’:

```
for i =1..p
  Find the best first order modification of column i and
  replace column i with it
end of for-loop
```

where ‘best’ above and ‘better’ below, of course, mean the modification that gives the best LH according to the chosen criterion. The name CP-sweep indicates that it is a systematic procedure going through (sweeping) all columns of the LH-matrix testing interchanges.

The complete CP algorithm can now be described in pseudo-code as follows:

```
Generate a random Latin hypercube: Lnew_
stop_ = FALSE
While( not stop_ )
  Lold_ = Lnew_
  Do a CP-sweep to generate Lnew_ from Lold_
  If stopping criterion fulfilled then stop_ = TRUE
End of while-loop
```

Concerning the stopping criteria in the while-loop, see Section 2.4.

Often in the text we write that we use an (random) LH, sometimes referred to as an RLH design of experiments. This is simply done by generating p random permutations of the numbers 1 to N and placing them as columns in the matrix. This procedure is much cheaper than, for example, a CP-sweep.

2.3. The genetic algorithm

The genetic algorithm is a very general optimization technique and can be applied to a large class of optimization problems. In the general form it can be described in pseudo-code as follows:

```
Generate initial population
Calculate fitness for individuals in the initial population
stop_ = FALSE
While( not stop_ )
  Select 'survivors'
  Cross-over the 'survivors'
  Mutate the resulting population
  Calculate the fitness of the new population
  If stopping criterion fulfilled then stop_ = TRUE
End of while-loop
```

The stopping criteria in the while-loop are discussed in Section 2.4.

There are many variations of the algorithm according to how one chooses to define the steps of selection, cross-over and mutation; also, the initial population can be chosen in various ways. For example, in Bates et al. (2003) a basic off-the-shelf genetic algorithm was employed for optimizing LHs with respect to criterion (3). However, this approach requires a special encoding of the design variable (which are points coordinates), and in order to enforce that the optimal design produces LH, the objective function must include term penalizing designs violating the concept of LH.

All the genetic operators that are used in this paper operate directly on $\boldsymbol{L}$ matrices (1) preserving the LH properties. They were designed particularly for the purpose of LH optimization. The major steps shown in the pseudo-code can be described for our problem as follows:

Initial population: We start by generating N_{pop} of random LHs that constitute the initial population. The number N_{pop} is required to be even because of the selection step.

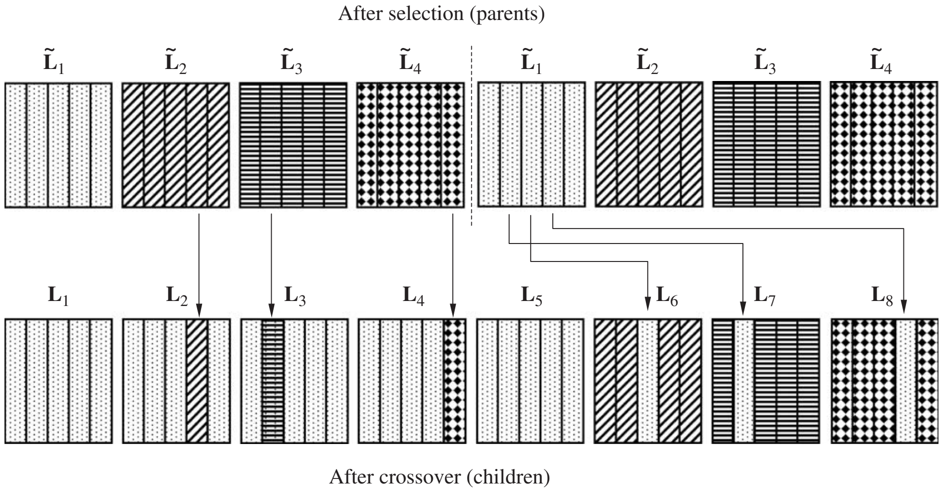

Fig. 1. Illustration of the cross-over step. Observe that after cross-over $\boldsymbol{L}_1$ and $\boldsymbol{L}_5$ will be identical to $\tilde{\boldsymbol{L}}_1$ before cross-over, the best LH. Later, in the mutation step, $\boldsymbol{L}_5$ is allowed to change, but not $\boldsymbol{L}_1$, so at least one copy of the best LH is always kept.

Selection: The $N_{\text{pop}}/2$ best LH are chosen as 'survivors' and the rest is thrown away.

Cross-over: In this step N_{pop} children of the $N_{\text{pop}}/2$ survivors are generated. This step is also illustrated in Fig. 1. First, the best LH is put as number 1 and $N_{\text{pop}}/2 + 1$ among the total set of children. Then the best LH is mated with the kth (for $k \in [2, N_{\text{pop}}/2]$) in the following way. The best and the kth will generate two children. The first child is obtained by taking the best LH and replacing a random column with the corresponding column in the kth LH. The resulting child is given the number k among the children. The second child is obtained by taking the kth LH and replacing a random column with the corresponding column in the best LH. The resulting child is given number $N_{\text{pop}}/2 + k$ among the children.

Mutation: Mutation is done on all except the best LH (the first child) from the earlier generation. The mutation of an LH is done in the following way. For each column, a random number in [0, 1] is generated (according to the uniform distribution) if this is lower than a threshold p_{mut}; then two randomly chosen elements in the column are swapped.

2.4. *Stopping criteria*

Here, only the case of the LH optimization with respect to the force criterion is considered. It will be shown in Section 4.1 that this criterion is a reasonable compromise between good statistical properties and efficiency. The stopping criteria for both the CP and the genetic algorithm are based on comparing the current improvement of the optimality criterion with the initial improvement. There are, however, some practical differences, and we start by describing the stopping criteria for the CP algorithm.

CP algorithm: The improvement in the first step, ΔG_1, is saved. Then in the kth step, the following inequality is checked:

$$\Delta G_k < \varepsilon \Delta G_1. \tag{4}$$

Here, ε is a chosen parameter. In the computations presented in this paper, the value $\varepsilon = 10^{-6}$ was used. If the inequality is satisfied the while-loop is ended. For example, to obtain an OLH with $N = 30$ and $p = 6$, 31 iterations in a test run are required. The value of the criterion for the resulting LH equals $G(\boldsymbol{L}) = 0.5331$.

Genetic algorithm: The step of producing one new generation in the genetic algorithm is cheap compared to the CP-sweep. Also, there is the possibility that the best LH in the new generation is not an improvement even if further iterations give important improvements. For these reasons, the accumulated improvement in the first n (say $n=50$ or 100) generations is saved (not only the first improvement):

$$\Delta\tilde{G}_n = G(\boldsymbol{L}_n) - G(\boldsymbol{L}_0), \tag{5}$$

where $\boldsymbol{L}_k$ is the best LH in the kth generation. Then in generation k, if k is a multiple of n, the following inequality is checked:

$$G(\boldsymbol{L}_k) - G(\boldsymbol{L}_{k-n}) < \varepsilon\Delta\tilde{G}_n. \tag{6}$$

In the computations, $\varepsilon = 10^{-7}$ was used. Applying this algorithm to compute an OLH with $N = 30$ and $p = 6$ OLH, approximately 3300 generations are required. The resulting LH have $G(\boldsymbol{L}) = 0.5326$.

2.5. Descriptive sampling

Here, we describe how to go from the discrete sampling matrix $\boldsymbol{L}$ determined by the OLH algorithm to the design of experiment matrix $\boldsymbol{X}$, where the distribution of each variable is taken into account.

We recall that each column in these matrices corresponds to one variable, which thus has a given probability distribution. In the matrix $\boldsymbol{L}$, a column is a permutation of the numbers 1 to N. To find the realization $x_k(m)$ of the random variable X_k, $1 \leqslant k \leqslant p$, corresponding to the number m, $1 \leqslant m \leqslant N$, in the kth column of the matrix $\boldsymbol{L}$ the cumulative distribution function (CDF) of X_k is used:

$$x_k(m) = F^{-1}(\tilde{x}_m), \tag{7}$$

where

$$\tilde{x}_m = \frac{m}{N} - \frac{1}{2N}. \tag{8}$$

In other words, the range of variability of each random variable is divided into N intervals of equal probability and the values $x_k(i)$, $i = 1, \ldots, N$ correspond to probabilistic midpoints (medians) of X_k in these intervals. Another choice, instead of (8), is to choose $x_k(i)$ randomly in the ith interval. However, in Kermat and Kielbasa (1999) and Huntington and Lyrintzis (1998) it was shown that the choice of medians or mean values (here a numerical integration is often required) of random variables in the intervals results in more precise estimators (smaller estimation variance).

It is important to mention that, in general, the random variables can be arbitrarily distributed and correlated. However, to use the sample design generated with LH, the variables must be first numerically transformed to a set of uncorrelated random variables. In the case when the joint probability density function is known, the Rosenblatt transformation (Rosenblatt, 1952) can be used, and when only marginal CDFs of variables and the correlation matrix are known, one may use the Nataf transformation (Nataf, 1962). Both transform the original variables to the space of independent standardized Gaussian variables. The values of the random variables found in the transformed space using (7) are next transformed back to the original random variables $\boldsymbol{X}$. Another approach of imposing correlation between input variables through reordering columns of $\boldsymbol{L}$ matrices was presented in Iman and Conover (1982) and Huntington and Lyrintzis (1998). However, in the present authors opinion, the transformation approach gives more flexibility since it does not require generation of an LH for each correlation structure of the input data. Instead, the pre-generated, data base OLH can be used for any correlations.

2.6. *Sobol sequences*

For all aspects of the generation of Sobol sequences we refer to Bratley and Fox (1988). That article describes the implementation of Algorithm 659 of ACM, which we have used to produce the results presented in Sections 4.1 and 4.4. Here, we only note that when the parameters of the method are fixed, as in the downloadable version (from Netlib) of Algorithm 659. Then, given N, the method produces an infinite “fixed” sequence of samples in the N-dimensional hypercube $[0, 1]^p$. To get N samples, we take the first N of these. Each coordinate of these can then be transformed according to the corresponding distribution as described in Section 2.5. The generation of Sobol sequences is less computationally expensive than OLH.

3. Analysis of the CP algorithm

Both the CP and the genetic algorithm seem impossible to analyze completely from the complexity point of view. However, for the CP, quite informative estimates for the execution time can be obtained. Below, some expressions on how the computational time depends on N, the number of samples and p, the number of variables, are presented.

Here, the term complexity is used to denote the asymptotic growth of execution time when N and p grow. For a function $T(N)$, the notation $T = \mathcal{O}(N^q)$ means that there are constants $\tilde{c}_1$ and $\tilde{c}_2$ such that

$$\tilde{c}_1 N^q < T < \tilde{c}_2 N^q, \tag{9}$$

for sufficiently large N. This fact will also be expressed by the formula $T \sim N^q$.

To determine the complexity of one CP-sweep, the following features of the algorithm must be taken into account:

- The outer loop is over the p columns.
- In each column, all first-order modifications are checked. There are $\mathcal{O}(N^2)$ such modifications.

- For each modification, the changed distances between the points must be updated. This requires $\mathcal{O}(Np)$ operations, see Section 5. When the distances have been computed, the criterion must be evaluated; this requires $\mathcal{O}(N^2)$ operations. For the execution time T_{crit} of this step we thus have

$$T_{\text{crit}}(N,p) \sim c_1 Np + c_2 N^2. \tag{10}$$

Summarizing the above information, the following complexity for the CP-sweep is obtained

$$T_{\text{CP}} \sim pN^2(c_1 Np + c_2 N^2). \tag{11}$$

The only remaining step in the analysis of the complexity of the CP-method is to determine how many times the (outer) while-loop is executed. We denote this (unknown) number by $\gamma(N,p)$. From our experimental tests, the following is a reasonable guess for the complexity of γ:

$$\gamma(N,p) \sim N. \tag{12}$$

Eqs. (11) and (12) together lead to the following expression for the execution time of the CP-algorithm:

$$T_{\text{tot}} \sim pN^3(c_1 Np + c_2 N^2). \tag{13}$$

It must be emphasized that the execution time is very sensitive to variations in N. As mentioned above, a number of tests have been performed to estimate the speed of the computations. These include the cases $N = 50, 60, \ldots, 100$ for $p = 3$, which indicate an execution time $T \sim N^q$ with q between 4.5 and 5.5. To illustrate the order of magnitude of the growth, we extrapolate from the execution time of approximately 3 min for the 100×3 OLH on an SGI Octane2 workstation, assuming $T \sim N^5$. This gives an execution time for 1000×3 OLH of 8 months.

4. Computational experiments

4.1. Comparison of criteria

In this section, two examples are investigated. In the first we study a function of two random variables and in the second, a mechanical model problem with six random variables.

The first test case consists of the so-called Rosenbrock function b defined as

$$b(X_1, X_2) = 100(X_2 - X_1^2)^2 + (1 - X_1)^2. \tag{14}$$

It is assumed that the two variables X_1 and X_2 are random and uniformly distributed in the interval [0, 2]. The goal is to obtain the mean value of the variable

$$Z = b(X_1, X_2), \tag{15}$$

which can be easily calculated analytically

$$\mathbb{E}(Z) = \int_0^2 \int_0^2 b(x_1, x_2)\,\frac{1}{4}\,\mathrm{d}x_1\,\mathrm{d}x_2 = 187. \tag{16}$$

Table 1
The average of the error percentage for the different sampling methods for different sample sizes

N	OLH Force	OLH Min. dist.	RLH	MC	Sobol
10	9.1	12.5	20.7	37.8	21.9
20	3.5	7.9	14.3	17.8	3.6
50	1.5	3.9	10.2	12.5	8.8
100	1.1	2.7	6.6	9.4	1.7
200	0.6	1.8	5.6	6.0	2.0
500	0.3	0.8	2.4	4.8	0.31
1000	—	—	1.5	3.2	0.01
2000	—	—	1.1	2.2	0.06
5000	—	—	0.7	1.2	0.03

OLH with N greater than 500 have not been computed because of the long computational time. The last column shows the error when the mean is computed with the Sobol sequence generator.

To evaluate the different sampling methods, we now set out to calculate this value with the following methods:

- OLH with the force criterion.
- OLH with the minimum distance criterion.
- A random Latin hypercube (i.e. without subsequent optimization), referred to as RLH.
- The 'standard' Monte Carlo (MC) method.
- Sobol sequences.

The results of the computations are shown in Table 1, where the average of the error percentage in the mean estimates is shown. To obtain these numbers, first, M designs of experiments $\{\boldsymbol{X}^{(k)}\}_{k=1}^{M}$ with the method in question are determined. The elements of these matrices are denoted by $x_{ij}^{(k)}$. Next, from these, M estimates $\overline{m}^{(k)}$ of the mean $\mathbb{E}(Z)$ are computed:

$$\overline{m}^{(k)} = \frac{1}{N} \sum_{i=1}^{N} b(x_{i1}^{(k)}, x_{i2}^{(k)}). \tag{17}$$

The value given in Table 1 is then the average of the error of these estimates, given as a percentage of the exact mean value,

$$\frac{1}{187M} \sum_{k=1}^{M} |\overline{m}^{(k)} - 187|. \tag{18}$$

For the Sobol sequences, the error using a single sequence, i.e. a single sampling, is shown in the table. This sequence is deterministic once several 'initial data' have been chosen; there is no natural way to chose these initial data randomly.

By examining the results in the table, it is easy to rank the first four methods. The best is the OLH optimized with the force criterion; then comes the OLH with the minimum distance criterion, the RLH and lastly MC which gives the largest average error in the mean estimation.

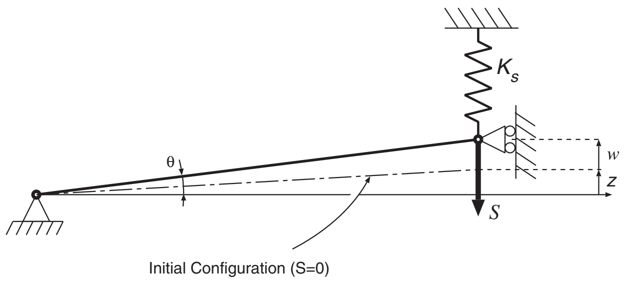

Fig. 2. The single bar structure prone to ‘snap-through’-type instability. The force S shown in the picture has negative sign.

Concerning Sobol, we note that, on the average, it is inferior to OLH with the force criterion for this example. We also observe its less regular convergence, with very good values for $N = 20$, 100, 500 and 1000, and rather poor values for $N = 50$, 200 and 2000.

The cost of generating large OLHs (with $N > 500$) grows fast. However, for the application we have in mind, which is stochastic simulation of car crash phenomenon, it is hardly imaginable one could afford 1000 or more sample points. The power of the OLH design is that it provides a sufficiently good quality of results with limited number of points.

The second test case, illustrated in Fig. 2, is a non-linear mechanical problem taken from Crisfield (1991, p. 2). The non-linearity results from the geometry; the constitutive relation for the bar and the spring are taken to be linear elastic. The unknown of the problem is w, the vertical displacement of the right node of the bar. The data are

- E: The Young modulus of the bar material.
- A: The cross-sectional area of the bar.
- l: The length of the unloaded bar.
- $z > 0$: The vertical coordinate of the right node of the bar when it is unloaded.
- K_s: The stiffness of the spring.
- S: The vertical force applied to the right node of the bar.

The displacement w can be determined from the following equation (cf. Crisfield, 1991):

$$S = \frac{EA}{l^3}\left(z^2 w + \frac{3}{2} z w^2 + \frac{1}{2} w^3\right) + K_s w. \tag{19}$$

In the derivation of this equation, it is assumed that the angle θ (see Fig. 2) is small, which implies $z, w \ll l$. Eq. (19) is a third-degree polynomial equation for w. For some values of the parameters there is one (unique) real root, for other values of the parameters there are three roots. In the case of three roots we choose the one corresponding to the smallest magnitude of displacement. This situation occurs when the applied force is smaller than the critical force that causes the bar to snap-through to the other equilibrium position. The case with one root of Eq. (19) corresponds to the state of the bar after snap-through or when the stiffness of the spring is so big that it prevents this kind of instability.

Table 2
The standard deviation for the mean estimates using different sample sizes N

N	OLH	RLH	MC
10	0.691	0.942	1.333
20	0.387	0.568	0.859
30	0.303	0.481	0.862
40	0.234	0.395	0.620
50	0.180	0.386	0.514
100	0.095	0.276	0.417
300	—	0.152	0.237
1000	—	0.087	0.151
3000	—	0.051	0.092

Now we turn to the stochastic description of the problem. The six variables are chosen to be uniformly distributed around the following mean values:

$$E = 5 \times 10^5\,\mathrm{N/mm^2},$$
$$A = 100\,\mathrm{mm^2},$$
$$l = 2500\,\mathrm{mm},$$
$$z = 25\,\mathrm{mm},$$
$$K_s = 0.9\,\mathrm{N/mm},$$
$$S = -22.5\,\mathrm{N}.$$

The interval of variation of the first five variables is taken to be 1% of the mean value. For the force we take the interval $[-23, -22]N$.

With the above distributions for the variables there is about 10% probability of snap-through behavior.

For this problem we also approximate the mean $\mathbb{E}(w)$ by the different sampling methods to compare their efficiency. Since the exact mean value is not known here, it is not possible to give the tables corresponding to those for the Rosenbrock function. From MC simulations we conclude that the mean is -17.923 ± 0.001. The results presented in Table 2 are instead the standard deviations of the mean estimates. Having computed M mean estimates $\overline{m}_k$, $k = 1, \ldots, M$ with a method, the standard deviation given in Table 2 is obtained by the following expression:

$$\sqrt{\frac{1}{M-1}\sum_{k=1}^{M}\left(\overline{m}_k - \frac{1}{M}\sum_{i=1}^{M}\overline{m}_i\right)^2}. \tag{20}$$

4.2. Determination of parameters in the genetic algorithm

Below we describe computational experiments to determine the values of the following parameters involved in the genetic algorithm:

- The population size N_{pop}.
- The probability of mutation p_{mut}.

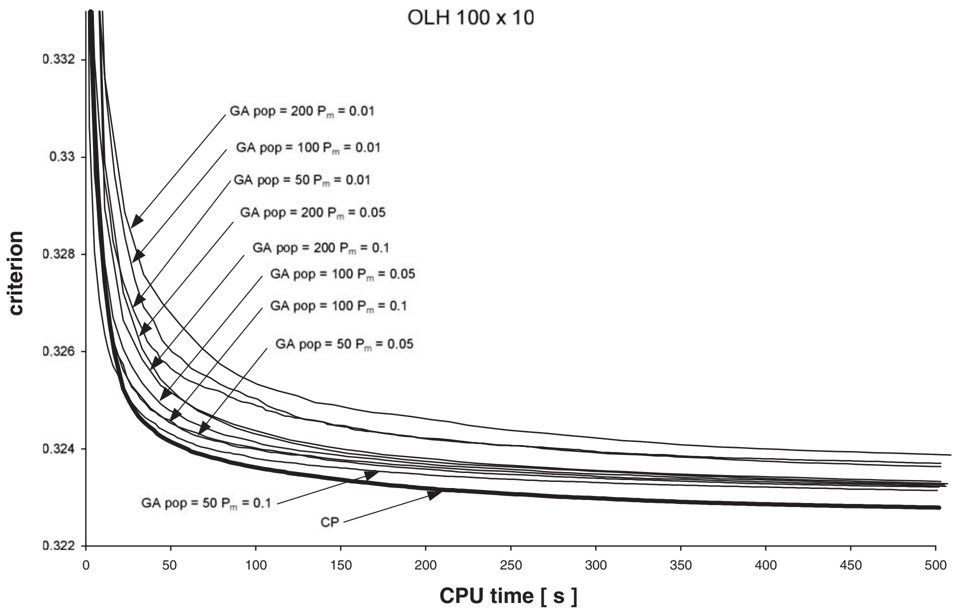

Fig. 3. Convergence curves for the CP algorithm and the genetic algorithm with various values of N_{pop} (denoted by GA pop on the graph) and p_{mut} (denoted by Pm) for an OLH with 100 points in 10 dimensions.

The optimal values of these parameters will depend on the number of sample points N and the number of parameters p of the LH.

We have done many investigations of the type illustrated in Figs. 3 and 4 for different sizes of LH ($N = 50, 100, \ldots, 300$, $p = 5, 10, \ldots, 30$). As was mentioned above, the optimal parameter values depend on the size of the problem. However, as a robust choice, we recommend $N_{\text{pop}} = 50$ and $p_{\text{mut}} = 0.1$, which has led to near optimal convergence in all the tests.

Concerning the comparison between the genetic algorithm and the CP algorithm, we advocate the use of CP for $N < 150$ and the genetic algorithm for larger problems. The boundary here is, of course, rather fuzzy and also depends slightly on p. The general trend, however, is clearly that the genetic algorithm gains compared to CP for larger problems. In particular its initial improvements are much bigger.

Another positive result is the robustness of the genetic algorithm. The convergence curves are always quite similar to those shown in Figs. 3 and 4. Furthermore, we have never experienced the potential pitfall of converging to a local minimum while CP finds a considerably better design.

4.3. Example OLH

In Fig. 5, we give some examples of LH with $N = 15$ and, the case that can be easily illustrated graphically, $p = 2$. It is difficult to identify 'visually' the superiority of the force

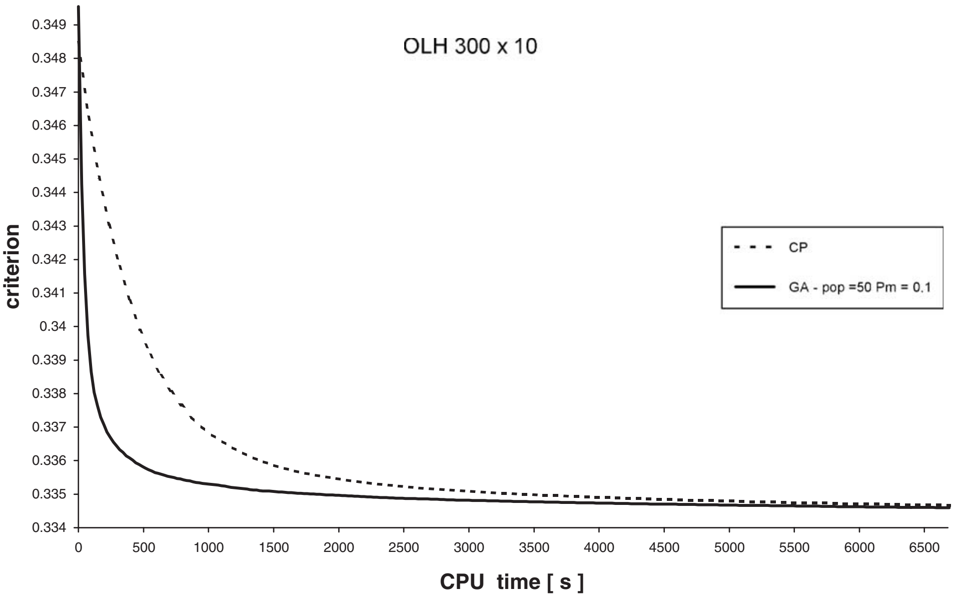

Fig. 4. Comparison of the convergence rate of the CP algorithm with the genetic algorithm for the near optimal choice of parameters $N_{\text{pop}} = 50$ and $p_{\text{mut}} = 0.1$. The size of the OLH is $N = 300$ and $p = 10$. The force criterion was used. It can be seen that for such a large problem the genetic algorithm is more effective.

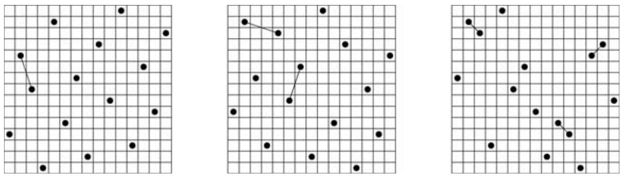

Fig. 5. Left: OLH with force criterion, center: OLH with minimum distance criterion, right: RLH. In the figures, the minimum distance between samples has been indicated with a line connecting the points in question. For the two OLH, the minimum distance is $\sqrt{10}$, and for the RLH, it is $\sqrt{2}$. Observe also that, by chance, the OLH optimized with the force criterion turn out to be better than the minimum distance OLH with respect to the *minimum distance* criterion (smaller $n(\boldsymbol{L})$). This is, however, a purely random phenomenon and depends on with which RLH the optimization starts.

criterion that was shown in Section 4.1. The samples of the RLH, however, are clearly more clustered and, consequently, leave certain areas ‘unexplored’.

4.4. The force criterion applied to Sobol sequences

In this section, we present computations of the force criterion $G(\boldsymbol{L})$, from Eq. (3), for Sobol sequences in a rather wide range of N- and p-values. Let $\{\boldsymbol{x}_i\}_{i=1}^{N} \subset [0, 1]^p$ denote the samples generated by Algorithm 659. To make the proper comparison with the OLH of

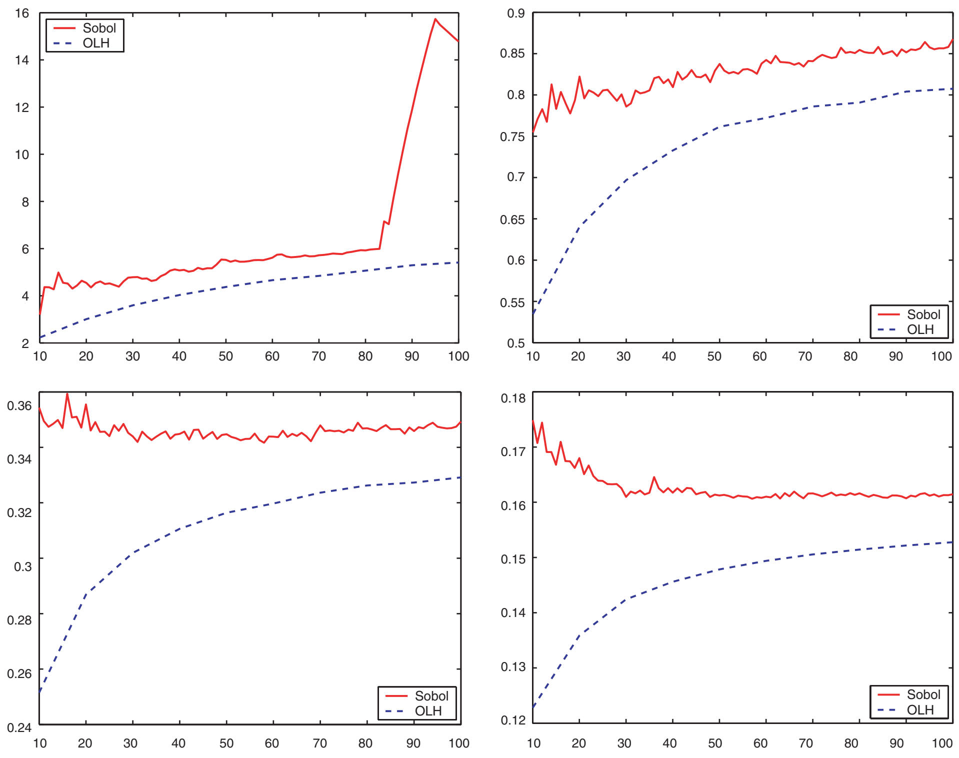

Fig. 6. Comparison of the force criterion for OLH and Sobol sequences. The dotted line represents the Sobol sequence and the full line OLH. On the abcissae we have N and the ordinate shows the value of $G(\boldsymbol{L})$. The upper left corresponds to $p = 2$, upper right $p = 5$, lower left $p = 10$, lower right $p = 20$.

the same size we then calculate

$$G(N\boldsymbol{L}),$$

i.e., we scale the matrix $L_{ik} = x_{ik}$ with N, the number of sample points, before evaluating the optimality criterion G. This is done for $p \in \{2, 5, 10, 20\}$ and N in the interval [10, 100]; the results are presented in Fig. 6. There it is seen that the OLH consistently gives a significantly better (lower) value of the optimality criterion than the Sobol sequence.

5. Remarks on the implementation

Especially concerning the CP method, there is a variety of implementation issues that strongly affects the efficiency of the program. Some are details concerning data structures, for example, the distance matrix, introduced below. In this section, however, we will only make a few remarks concerning the following issues in the organization of the computation:

- No square root in the criterion calculation.
- Update only needed for elements when interchanging.

Square root: When evaluating the function $d(\boldsymbol{L})$, see Eq. (2), one must first calculate all the distances. We denote the distance between point $\mathbf{x}_i$ and point $\mathbf{x}_j$ by d_{ij}. Then all these numbers are searched for the minimum. It is, however, more efficient to compute and compare d_{ij}^2 since this does not require the square root computation. This may not seem important, but the criterion is evaluated in the innermost loop of the CP method and many (see next point) distance calculations must be done.

Update: Except in the first iteration, we do not need to update the entire distance matrix between the evaluations of the criteria. In between two evaluations only two elements of the $\boldsymbol{L}$ matrix are interchanged; thus, it is necessary only to update two columns and two rows in the distance matrix. Of course, since the distance matrix is symmetric, only the half of it (without diagonal) must be stored. Thus, $2N - 1$ elements in the distance matrix are updated between each first-order modification of the CP method.

6. Conclusions

The design of experiment is a crucial component of stochastic simulation of complex, computationally expensive problems.

For problems where one can afford a medium number of samples (say 20–500), the use of optimized LH is recommended.

Based on the computations presented in Section 4.1, the use of the force optimality criterion for LH optimization is recommended over the minimum distance criterion.

Depending on the size of the LH, the CP algorithm or a genetic algorithm is found to be most competitive. Our experiments, as described in Section 4.2, suggest the use of CP for N less than 150 and the genetic algorithm for larger problems. We furthermore give a complete description of some new modifications of the genetic algorithm for this problem. The description includes optimal values for the parameters in the algorithm.

The results in Sections 4.1 and 4.4 indicate that OLH can be an alternative to Sobol sequences for mean prediction (and thereby for integration). This statement must, of course, be seen in the context of this paper: computationally expensive problems with a medium number of samples affordable.

Acknowledgements

This work has been supported by Marie Curie Fellowship of the European Community programme GROW under contract number G3TR-CT-2000-00038. This support is gratefully acknowledged. The author, R. Stocki, would also like to gratefully acknowledge the partial support from The Foundation of Polish Science (FNP no. 4/2001).

References

Audze, P., Eglais, V., 1977. New approach to planning out of experiments. Problems Dyn. Strength 35, 104–107 (in Russian).

Bates, S.J., Sienz, J., Langley, D.S., 2003. Formulation of the Audze–Eglais uniform Latin hypercube design of experiments. Adv. Eng. Software 34, 493–506.

Braibant, V., Bulik, M., Liefvendahl, M., Molinier, S., Stocki, R., Wauquiez, C., 2002. Stochastic simulation of highly nonlinear dynamic systems using the M-XPLORE extension of the RADIOSS software. In: Workshop on Optimal Design of Materials and Structures, Ecole Polytechnique (on CD).

Bratley, P., Fox, B.L., 1988. Algorithm 659. implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Software 14 (1), 88–100.

Caflisch, R.E., 1998. Monte Carlo and quasi-Monte Carlo methods. Acta Numer. 7, 1–49.

Crisfield, M.A., 1991. Non-linear Finite Element Analysis of Solids and Structures, vol. 1. Wiley, New York.

Huntington, D.E., Lyrintzis, C.S., 1998. Improvements to and limitations of Latin hypercube sampling. Probab. Eng. Mech. 13, 245–253.

Iman, R.L., Conover, W.J., 1982. A distribution-free approach to inducing rank correlation among input variables. Comm. Statist. 11, 311–334.

Johnson, M.E., Moore, L.M., Ylvisaker, D., 1990. Minimax and maximin distance designs. J. Statist. Plann. Inference 26, 131–148.

Kermat, M., Kielbasa, R., 1999. Modified Latin hypercube sampling Monte Carlo (MLHSMC) estimation for average quality index. Analog Integrated Circuits Signal Process. 19, 87–98.

McKay, M.D., Beckman, R.J., Conover, W.J., 1979. A comparison of three methods for selecting values of input variables from a computer code. Technometrics 21, 239–245.

Mitchell, T.J., 1974. Computer construction of *d*-optimal first-order designs. Technometrics 16, 211–220.

Morris, M.D., Mitchell, T.J., 1995. Exploratory designs for computer experiments. J. Statist. Plann. Inference 43, 381–402.

Nataf, A., 1962. Determination des distribution dont les marges sont donnees. C.R. Acad. Sci. 225, 42–43.

Niederreiter, H., 1992. Random number generation and quasi-Monte Carlo methods. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63. SIAM, Philadelphia.

Park, J.-S., 1994. Optimal latin-hypercube designs for computer experiments. J. Statist. Plann. Inference 39, 95–111.

Rosenblatt, M., 1952. Remarks on a multivariate transformation. Ann. Math. Statist. 23, 470–472.

Simpson, T.W., 1998. A concept exploration method for product family design. Ph.D. Thesis, Georgia Institute of Technology.

Ye, K.Q., Li, W., Sudjianto, A., 2000. Algorithmic construction of optimal symmetric Latin hypercubes. J. Statist. Plann. Inference 90, 145–159.