Sequential design for prediction Sequential design for prediction with Gaussian process models

Mona Abtini, Céline Helbert, François Musy, Luc Pronzato, Maria-João Rendas

Sequential design for prediction with Gaussian process models

Mona Abtini mona.abtini@doctorant.ec-lyon.fr,
Céline Helbert celine.helbert@ec-lyon.fr,
François Musy francois.musy@ec-lyon.fr
ECL, ICJ, UMR 5208
Université de Lyon
36 av. G. de Collongue
69134 Ecully Cedex, FRANCE

Luc Pronzato pronzato@i3s.unice.fr
Maria-João Rendas rendas@i3s.unice.fr
I3S - UMR7271
CNRS-UCA
2000 route des lucioles
06900 Sophia Antipolis, FRANCE

Editor: tbd

Abstract

When numerical simulations are time consuming, the simulator is replaced by a simple (meta-)model which approximates its behavior. This surrogate model is adjusted on a set of carefully chosen computer experiments, or Design of Experiments (DoE), with the objective of obtaining the best possible approximation with available computational budget. For the widely used Gaussian process models, Mutual Information (MI) is a particularly attractive DoE quality measure. Finding the set of configurations that optimise the MI criterion is a NP-hard problem, and in this paper we concentrate on the sequential construction of designs by greedily maximising MI. Unfortunately, even this much simpler problem is still computationally demanding, still involving prohibitively large running times. We propose a new algorithm for the sequential construction of MI-optimal designs that shares computational costs across several candidate points, enabling a significant reduction of computing time. In particular, we show that the combination of our approach with a Lazy-greedy strategy proposed previously leads to important computational gains, enabling the consideration of more challenging problems (higher dimensional problems, finer grids of design points). A comprehensive numerical study highlights the increased invariance of the computational costs of the new algorithm with respect to implementation choices, like the covariance kernel.

Keywords: Design of Experiments, Mutual Information, Computational Complexity

1. Introduction

Computer experiments have become an instrumental alternative to real experiments in the study of physical phenomena. However, analysis of complex systems (in particular high-dimensional problems, which involve many input variables) requires in general a significant

number of time-consuming simulations, each one possibly taking from several hours to several days. A Design of Experiments (DoE) specifying a small number N of carefully chosen simulations allows the identification of an accurate (meta-)model of the simulator which can be used to concentrate subsequent analysis in the regions of the input domain potentially relevant to the goals of the study. In this paper, we consider meta-models that assume that the observed system is a realisation of a Gaussian process.

In the framework of Gaussian process modelling, Mutual Information (MI, Caselton and Zidek (1984)) is used to quantify the decrease of uncertainty about unobserved values of the process that results from using a given design. Maximizing the MI between the observations at the design points and the function values over the rest of its domain should thus lead to meta-models with good predictive quality. As for many other performance criteria, the direct determination of the N-point design with maximal MI is a difficult (NP-hard, non-linear, multivariate) problem. In this paper we focus on using sequential greedy optimisation to find DoEs with near-optimal MI.

Definition 1 (greedy optimisation) *Let ϕ be a criterion defined on the power set of a finite set $\mathbf{D}_M$. A greedy algorithm for the maximisation of ϕ, initialized at $A_0 = \emptyset$ (the empty set), sequentially selects points in $\mathbf{D}_M$ according to*

$$x_{n+1} \in \underset{x \in \mathbf{D}_M \setminus A_n}{\operatorname{Arg\,max}} \, \phi\left(A_n \cup \{x\}\right) \, , \; A_{n+1} = A_n \cup \{x_{n+1}\} \, .$$

Unless for additive criteria, the quality of a DoE obtained by the greedy construction above cannot in general be guaranteed. However, because the MI criterion is submodular and quasi-monotonic, see Krause et al. (2008), the MI of the solutions A_n obtained at all iterations ($n = 1, 2, \ldots$) of the greedy algorithm can be proved to achieve at least 63% of its maximum possible value, as a consequence of a celebrated theorem of Nemhauser et al. (1978).

However, even this sequential construction of designs with guaranteed MI may remain prohibitive when the size of $\mathbf{D}_M$ is large, since it requires the inversion of a correlation matrix of size $|\mathbf{D}_M \setminus A_n| - 1$. As shown in Krause et al. (2008), lazy-greedy optimisation, see Minoux (1978), which avoids in each iteration actual computation of MI over the entire set $\mathbf{D}_M \setminus A_n$, can be used to decrease the overall computational time, submodularity guaranteeing no subsequent loss of performance. The number of evaluations of the criterion performed at each iteration can thus be decreased. Unfortunately, the numerical complexity of each evaluation of $\phi(\cdot)$ may still remain prohibitively large when $|\mathbf{D}_M|$ is very large.

This paper addresses this last curb on use of MI as a design criterion, presenting an efficient implementation of the MI evaluations performed at each iteration of the greedy algorithm, which must consider possible extensions $A_n \cup \{x\}$, for $x \in \mathbf{D}_M \setminus A_n$, of the current design A_n. The gain of our implementation is based on a partition of the set of remaining candidate points, $\mathbf{D}_M \setminus A_n$, which enables the mutualisation of certain matrix inversions, being an example of the familiar trade-off between memory requirements and computational complexity. We show that this efficient evaluation, based on block decomposition of the correlation matrices, can be combined with the idea proposed in Krause et al. (2008) to further improve the overall efficiency of design construction. Contrary to the Stochastic-Greedy algorithm presented in Mirzasoleiman et al. (2015), the improvement in

computational complexity proposed here does not come at the price of performance degradation.

The structure of the paper is as follows. Section 2 recalls the definition of Mutual Information and remembers its expression under the Gaussian process assumption. In Section 3 we present briefly the sequential construction method proposed in Krause et al. (2008) (Algorithm Lazy). Our new algorithm based on block partition (Algorithm Blocks) is described in Section 4, where the influence of the number of blocks on the computing time is investigated and a rule for its quasi-optimal choice given. In Section 5 we combine both ideas, lazy-greedy and block partition (Algorithm Lazy-Blocks), and present a numerical comparison of the complexity of the three algorithms. Details on the analysis of the complexity of Algorithm Blocks, which resorts to a more exact characterisation of the numerical complexity of Cholesky factorisation than what can easily be found in the literature, are collected in Appendix A.

2. Mutual information in Gaussian process modelling

In this section we recall the principles of Gaussian process regression, the definition of Mutual Information, and its analytical expression for Gaussian process models.

2.1 Gaussian process modelling

Let D be a compact subset of a d-dimensional space and $\mathbf{D}_M$ an M-point discretization of D, for example, when $D = [0,1]^d$, $\mathbf{D}_M$ can be a regular grid or the first M points of a low discrepancy sequence. Let $X \subset \mathbf{D}_M$ be a DoE with N points. We write $X = (x_i^j)_{1\leq i\leq N, 1\leq j\leq d}$, where x_i^j is the j-th coordinate of the i-th point and denote $\overline{X} = \mathbf{D}_M \setminus X$.

Gaussian process regression assumes that the response of the simulator to inputs $x \in D$ is a realization of a Gaussian process (see Rasmussen and Williams (2005)) $(Y(x))_{x\in D}$, with known mean γ, where $\gamma : x \in D \to \gamma(x) \in \mathbb{R}$, and covariance $k : (x_i, x_k) \in D \times D \to k(x_i, x_k) = \sigma^2 r(x_i, x_k) \in \mathbb{R}$, where $r(.,.)$ is a correlation kernel. After observation of the responses of the simulator on the DoE, $Y_X = \{Y(x), x \in X\}$, a meta-model approximates the simulator response on D by the expected value of the posterior Gaussian process $((Y(x))_{x\in D} \mid Y_X) \sim GP_Y(\mu_{x|X}, \sigma^2_{x|X})$, where

$$\mu_{x|X} = \mathbb{E}[Y(x)|Y_X] = \gamma(x) + r_{x,X}^t R_{X,X}^{-1}(Y_X - \gamma(X)), \quad (1)$$

$$\sigma^2_{x|X} = \mathbb{V}[Y(x)|Y_X] = \sigma^2(1 - r_{X,x}^t R_{X,X}^{-1} r_{X,x}). \quad (2)$$

Above, we used notation $R_{X_1,X_2} = (r(x_i, x_j))_{x_i\in X_1, x_j\in X_2}$, with X_1 and X_2 two DoE's (having possibly different sizes), $r_{X,x} = (r(x_i, x))_{1\leqslant i\leqslant |X|}$, $\gamma(X) = (\gamma(x_i))_{1\leqslant i\leqslant |X|}$ and $r_{X,x}^t$ is the transpose of $r_{X,x}$. In (1,2), $R_{X,X}$ is thus a $|X| \times |X|$ symmetric positive-definite matrix and $r_{X,x}$ and $\gamma(X)$ are columns vectors of length $|X|$. The mean function $\gamma(x)$ does not play any role in MI-based construction of DoE, and we will assume without loss of generality that $\gamma(x) = 0$ for all $x \in D$.

2.2 The MI criterion

We present below basic definitions from information theory. All the material can be found in classic references, see for instance Cover and Thomas (1991). We start by the

elemental definition of Shannon entropy of a random variable $X \sim g(x)$ as $H(X) = -\int_{\mathbb{R}} g(x) \log g(x)\, dx$ For jointly distributed random variables X and Y, $(X, Y) \sim g(x, y)$ let $Y \sim g_Y(y)$, $X \sim g_X(x)$ and $g_{X|y}(x|y)$ denote the conditional density of x given $Y = y$. The conditional (Shannon) entropy of X given Y is by definition $H(X|Y) = -\int_{\mathbb{R}} g_Y(y) \int_{\mathbb{R}} g_{X|y}(x|y) \log g_{X|y}(x|y)\, dx\, dy$.

Definition 2 (mutual information) *The mutual information between a pair of random variables (X, Y) is defined by:*

$$\mathsf{MI}(X;Y) = \int_{\mathbb{R}} \int_{\mathbb{R}} g(x,y) \log \left[\frac{g(x,y)}{g_X(x) g_Y)(y)} \right] \mathrm{d}x\mathrm{d}y \,.$$

$\mathsf{MI}(X;Y)$ measures the degree of statistical dependency between the two random variables X and Y, and $\mathsf{MI}(X;Y) = 0$ if and only if they are statistically independent. In fact, mutual information measures the decrease of the Shannon entropy of one of the variables when the other is observed:

$$\begin{aligned} \mathsf{MI}(X;Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X)\,, & (3)\\ &= H(X) + H(Y) - H(X,Y)\,, & (4) \end{aligned}$$

the last equation resulting from the chain rule of entropy $H(X, Y) = H(X|Y) + H(Y)$.

In the context of DoE, one should choose X such that the mutual information $\mathsf{MI}(Y_X; Y_{\overline{X}})$ between observed (Y_X) and unobserved ($Y_{\overline{X}}$) simulator outputs is maximal. By (4),

$$\mathsf{MI}(Y_X; Y_{\overline{X}}) = H(Y_X) - H(Y_{\mathbf{D}_M}) + H(Y_{\overline{X}})\,. \tag{5}$$

If Y_X and $Y_{\overline{X}}$ are jointly Gaussian (which is the case, under the Gaussian process assumption, when Y_X gathers the observed values of the simulator response) the Shannon entropy is given by

$$H(Y_X) = \frac{N}{2}\left[1 + \log(2\pi\sigma^2)\right] + \frac{1}{2}\log(\det R_{X,X})\,, \tag{6}$$

and the mutual information criterion $\mathsf{MI}(Y_X; Y_{\overline{X}})$ is given by

$$\mathsf{MI}(Y_X; Y_{\overline{X}}) = \frac{1}{2}\left[\log(\det R_{X,X}) - \log(\det R_{\mathbf{D}_M,\mathbf{D}_M}) + \log(\det R_{\overline{X},\overline{X}})\right]\,. \tag{7}$$

For simplicity, in the sequel we shall write $H(X)$ for $H(Y_X)$ and $\mathsf{MI}(X)$ for $\mathsf{MI}(Y_X; Y_{\overline{X}})$. Note that it is an immediate consequence of (5) that $\mathsf{MI}(\emptyset) = 0$.

3. Greedy MI maximisation

In general computing an MI-optimal N-point design is a very difficult and time-consuming task: the optimisation problem is high dimensional, since $N \times d$ coordinates must be determined simultaneously, and several local optima exist in general. Although a sequential greedy construction – see Definition 1 – leads to a suboptimal solution, it has two important advantages: first, at each step the optimisation concerns only one (d dimensional) design point; second, and equally important, the final size of the design does not need to be fixed in advance.

Moreover, as we remember in subsection 3.1 below, it has been proved that under convenient assumptions on the optimised criterion it is possible to provide guarantees on the quality of the designs identified at each step of the sequential construction.

3.1 Submodularity

Let A be a finite set. The power set of A, usually denoted by 2^A, is the set of all subsets of A. Naturally, $|2^A| = 2^{|A|}$.

Definition 3 (submodularity Bach (2011)) *A function ϕ defined on the power set of $\mathbf{D}_M$ is submodular if and only if it satisfies the following two (equivalent) conditions:*

$$\text{for all subsets } A \text{ and } B \text{ of } \mathbf{D}_M\,, \qquad \phi(A) + \phi(B) \geq \phi(A \cup B) + \phi(A \cap B)\,, \tag{8}$$

$$\text{for all } A \subset B \subset \mathbf{D}_M \text{ and } x \in \mathbf{D}_M \setminus B\,, \qquad \phi(A \cup \{x\}) - \phi(A) \geq \phi(B \cup \{x\}) - \phi(B)\,. \tag{9}$$

For those unfamiliar with the notion of submodularity, equation (9) provides an intuitive meaning of this concept as the celebrated "diminishing returns" property: the increase in the value of ϕ when an element is added to a set A is a decreasing function of this set.

The following definition will prove useful in the sequel.

Definition 4 (efficiency) *Let ϕ be a criterion defined on the power set of $\mathbf{D}_M$ that should be maximised.*

The efficiency of $A \subset \mathbf{D}_M$ is the ratio $\phi(A)/\phi^\star_{|A|}$, where $\phi^*_N = \max_{A \subset \mathbf{D}_M, |A|=N} \phi(A)$ denotes the optimal criterion value for sets of size N.

Theorem 1 (guaranteed efficiency, Nemhauser et al. (1978)) *Let $\phi(\cdot)$ be an increasing submodular criterion defined on the powerset of $\mathbf{D}_M$ for which $\phi(\emptyset) = 0$. Let A_N be the set of the N first points selected by the greedy algorithm and denote $\mathrm{e} = \exp(1)$, then:*

$$\frac{\phi(A_N)}{\phi^*_N} \geq \left(1 - \frac{1}{\mathrm{e}}\right) \simeq 0.63\,.$$

The theorem states that, provided that the criterion satisfies suitable conditions, the greedy construction guarantees an efficiency of at least 63%, whatever the size N of the design constructed ($N \leq M$). Note that the selection rule indicated in Definition 1 is equivalent to maximisation of $\delta_{A_n}(x)$, the increment of ϕ at A_n:

$$\operatorname*{Arg\,max}_{x \in \mathbf{D}_M \setminus A_n} \phi\left(A_n \cup \{x\}\right) = \operatorname*{Arg\,max}_{x \in \mathbf{D}_M \setminus A_n} \delta_{A_n}(x), \qquad \text{where} \qquad \delta_X(x) = \phi(X \cup \{x\}) - \phi(X)\,.$$

If ϕ is submodular these increments are, for each x, decreasing functions of A_n, that is, of the iteration n.

3.2 Greedy construction for the MI criterion

As noted before, MI trivially verifies the normalisation condition $\phi(\emptyset) = 0$. On the contrary, $MI(X) \equiv MI(Y_X; Y_{\overline{X}})$ is symmetric in its two arguments and thus strictly speaking the mutual information criterion cannot be a monotone function, violating one of the conditions of Theorem 1. Nevertheless, for small design sizes $N \leq \lfloor M/2 \rfloor$ the MI criterion can be seen to satisfy the conditions of Theorem 1, see Krause et al. (2008), and thus the efficiency of the DoEs obtained by greedy maximisation can be guaranteed for $N < \lfloor M/2 \rfloor$.

The increment $\delta_X(x)$ when adding point x to X can be expressed in terms of conditional Shannon entropies (see (5))

$$\delta_X(x) = \mathsf{MI}(X \cup \{x\}) - \mathsf{MI}(X) = H(\{x\}\,|X) - H(\{x\}\,|\overline{X} \setminus \{x\}), \qquad x \in \overline{X}\,. \tag{10}$$

Note that $\delta_X(x) = 0$ when $x \in X$.

Under the Gaussian assumption, see equations (6) and (2), each conditional entropy is of the form

$$H(\{x\}\,|Z) = \frac{1}{2}\,\left[1 + \log(2\pi\sigma^2) + \log\left(1 - r_{Z,x}^t R_{Z,Z}^{-1} r_{Z,x}\right)\right]\,,$$

for either $Z = X$ or $Z = \overline{X} \setminus \{x\}$. The costly operation in the expression above is the inversion of $R_{Z,Z}$, which is $O(M^3)$. While for the first term of (10), for which $Z = X$, this is a small matrix of size N, for the second term $|Z| = |\overline{X} \setminus \{x\}\,| = M - N - 1$ is in general very large.

At each iteration one such matrix inversion must be done for each element of $\overline{X} = \mathbf{D}_M \setminus X$, whose cardinality is $O(M)$ when $N \ll M$, leading to a unrealistic complexity $O(NM^4)$ for choosing a quasi-MI-optimal design of size N if a straightforward implementation of the greedy algorithm is used. In Section 4.1 we present an efficient numerical implementation that shares some of the computational load amongst evaluations for distinct $x \in \mathbf{D}_M \setminus X$, by relying upon a suitable partition of $R_{\overline{X},\overline{X}}$.

3.3 Lazy-greedy: exploiting the submodularity of MI

In Krause et al. (2008) the authors show that submodularity of the MI criterion allows its lazy-greedy optimisation, avoiding exhaustive search through all remaining candidate points at each greedy iteration. Computational time can thus be reduced without sacrificing the original performance guarantee.

In fact, since MI is submodular it thus verifies the "diminishing returns" property,

$$\text{for all } X_k \subset X_{k+1} \subset \mathbf{D}_M \text{ and for all } x \in \overline{X}_{k+1},, \delta_{X_{k+1}}(x) \leq \delta_{X_k}(x)\,.$$

This means that increments computed in previous iterations of the greedy algorithm are upper bounds on the value they can take in future iterations. Algorithm Lazy presented below, proposed in Krause et al. (2008), outputs the N point design that is chosen by greedy optimisation of MI. During the iterations, it maintains in memory an upper bound $(\overline{\delta}(x))$ on the value of the actual MI increment with respect to the current design $(\delta_{X_n}(x))$ for each $x \in \mathbf{D}_M \setminus X_n$, that is,

$$\overline{\delta}(x) \geq \delta_{X_n}(x), \qquad \forall x \in \mathbf{D}_M \setminus X_n\,.$$

The computational gains of Algorithm Lazy arise from direct placement of elements x with $\overline{\delta}(x_i) \leq \delta(x_*)$ in the list $\mathcal{L}_n$ without actual update of their increment at that iteration, and is difficult to quantify.

While the worst case complexity of Algorithm Lazy is still $O(NM^4)$, the acceleration observed in real applications is often very large, as we will see in Section 5.2. In Mirzasoleiman et al. (2015) the authors propose a stochastic version of this algorithm that is based on update of only a random sample of the set of candidate points that has complexity linear in the size of the domain. However, this acceleration comes at the price of a decreased performance guarantee of $1 - 1/e - \epsilon$, where ϵ is determined by the size $((M/N)\log(1/\epsilon))$ of the subset of candidate points that is actually updated in each iteration.

Algorithm A (Algorithm Lazy)

for all $x \in \mathbf{D}_M$ **do**
  $\overline{\delta}(x) = MI(\{x\})$;
end for
$x_1 = \arg\max_x \overline{\delta}(x)$, $X_1 = \{x_1\}$; set $\overline{\delta}(x_1) = 0$;
for $n = 2$ to $N \leq \lceil M/2 \rceil$ **do**
  include all $x \in \mathbf{D}_M$ with $\overline{\delta}(x) > 0$ in the candidate list $\mathcal{L}_n$, sorted by decreasing values of $\overline{\delta}(x_i)$;
  while $|\mathcal{L}_n| > 1$ **do**
    let x_* be the first x in $\mathcal{L}_n$, compute $\delta(x_*)$ and update $\overline{\delta}(x_*) = \delta(x_*)$;
    remove from $\mathcal{L}_n$ all the x_i such that $\overline{\delta}(x_i) \leq \delta(x_*)$;
    place x_* in the right position in $\mathcal{L}_n$ (sorted by decreasing values of $\overline{\delta}(x_i)$);
  end while
  Take x_n has the only element in $\mathcal{L}_n$; set $X_n = X_{n-1} \cup \{x_n\}$;
end for

4. Partitioning

We present now the major result of the paper, showing how the overall complexity of each greedy iteration can be decreased by working with a partition of the set of candidate points, addressing also the determination of its optimal size.

4.1 Principle

By equation (7), maximizing $\mathsf{MI}(X \cup \{x\})$, where $x \in \overline{X}$, is equivalent to maximizing the sum

$$\delta(x) = \log(\det R_{X\cup\{x\},X\cup\{x\}}) + \log(\det R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}}) \ . \tag{11}$$

At each greedy iteration we must compute the two terms in the left handside for each candidate point x in $\overline{X}$. This section proposes a method that decreases the computational load of each greedy iteration by mutualizing the some of the numerical load across the blocks of a partition of $\overline{X}$.

Consider the addition of x to the current design X and write $R_{X\cup\{x\},X\cup\{x\}}$ as

$$R_{X\cup\{x\},X\cup\{x\}} = \begin{bmatrix} R_{X,X} & r_{X,x} \\ r_{X,x}^t & 1 \end{bmatrix} ,$$

such that $\log(\det R_{X\cup\{x\},X\cup\{x\}}) = \log(1 - r_{X,x}^t R_{X,X}^{-1} r_{X,x}) + \log(\det R_{X,X})$. Using the Cholesky factorization $R_{X,X} = S^t S$, we obtain

$$\log(\det R_{X\cup\{x\},X\cup\{x\}}) = \log\left[1 - (S^{-t} r_{X,x})^t (S^{-t} r_{X,x})\right] + 2\log(\det S)\,. \tag{12}$$

Matrix S does not depend on x, and thus only $S^{-t} r_{X,x}$ needs to be computed for each new candidate point x.

Although less simple, it is also possible to identify computations involved in the evaluation of $\log(\det R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}})$ that can be shared amongst distinct candidate points x, by considering a partition of the set of all candidate points $\overline{X}$.

Let $L \subset \overline{X}$, $\ell \in L$ a generic element of L, and partition $\overline{X}$ as

$$\overline{X} = I \cup L^{-\ell} \cup \{\ell\}\,, \qquad \text{where } \; I = \overline{X} \setminus L, \qquad \text{and } \; L^{-\ell} = L \setminus \{\ell\}\,. \tag{13}$$

Write —if required by applying a suitable permutation of its lines and columns— $R_{\overline{X},\overline{X}}$ in the corresponding block form:

$$R_{\overline{X},\overline{X}} = \begin{bmatrix} R_{I,I} & R_{I,L^{-\ell}} & R_{I,\ell} \\ R^t_{I,L^{-\ell}} & R_{L^{-\ell},L^{-\ell}} & R_{L^{-\ell},\ell} \\ R^t_{I,\ell} & R^t_{L^{-\ell},l} & 1 \end{bmatrix}.$$

We must calculate the log-determinant of this matrix when its last line and column are deleted:

$$\det R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}} = \det \begin{bmatrix} R_{I,I} & R_{I,L^{-\ell}} \\ R^t_{I,L^{-\ell}} & R_{L^{-\ell},L^{-\ell}} \end{bmatrix} = \log(\det R_{I,I}) + \log(\det(B_\ell))\,, \tag{14}$$

where B_ℓ is the Schur complement of block $R_{I,I}$

$$B_\ell = R_{L^{-\ell},L^{-\ell}} - R^t_{I,L^{-\ell}} R^{-1}_{I,I} R_{I,L^{-\ell}}\,. \tag{15}$$

We see that the term $\log\det(R_{I,I})$ in (14) is common to all $\ell \in L$ showing that the Cholesky factorization $R_{I,I} = S^t_L S_L$ and the determination of its determinant can be shared amongst all elements of block L.

Besides, for each different ℓ we need to compute

$$B_\ell = R_{L^{-\ell},L^{-\ell}} - D_{L^{-\ell},L^{-\ell}}, \qquad \text{where} \qquad D_{L^{-\ell},L^{-\ell}} = R^t_{I,L^{-\ell}} R^{-1}_{I,I} R_{I,L^{-\ell}}\,. \tag{16}$$

Note that B_ℓ can be obtained by deleting both its ℓ-th line and column of matrix $B_{L,L}$:

$$B_\ell = [B_{L,L}]^{\sim\ell,\sim\ell}\,, \qquad \text{where } \; B_{L,L} = R_{L,L} - D_{L,L}, \qquad \text{and} \qquad D_{L,L} = R^t_{I,L} R^{-1}_{I,I} R_{I,L}\,. \tag{17}$$

Finally, matrix $D_{L,L}$ is computed by using the Cholesky factorisation of $R_{I,I}$, as

$$D_{L,L} = C^t_L C_L, \qquad \text{with} \qquad C_L = S^{-t}_L R_{I,L}\,. \tag{18}$$

We presented above how considering a binary partition $\overline{X} = I \cup L$ of the set of candidate points enables to share computations amongst the points belonging to each element of either L or I (remark that the derivation above is symmetric in sets I and L). More generically, the set of candidate points can be partitioned in m subsets, $\overline{X} = \cup_{v=1}^m L_v$, and computations shared amongst the elements of each subset L_v as we described above. The overall organisation of the computations is summarised in Algorithm Blocks below.

4.2 Complexity of Algorithm Blocks

The complexity of Algorithm Blocks depends heavily on the number m of blocks into which $\overline{X}$ is partitioned: the number of required factorizations of $R_{I,I}$ increases with m, while the computational cost of the factorization of B_ℓ must decrease with m since the size of each block L_v is smaller. An optimal balance between these two terms must thus exist, and

Algorithm B (Algorithm Blocks)

1: {**Initialization**}
   $X = \emptyset$,
2: **for** $n = 1$ to N **do**
3: Perform Cholesky factorization $R_{X,X} = S^t S$
4: **for** all $\ell \in \overline{X}$ **do**
5: Compute $\log\det(R_{X\cup\{x\},X\cup\{x\}})$ (eq. (12))
6: **end for**
7: Partition $\overline{X} = \bigcup_{v=1}^{m} L_v$
8: **for** $v = 1$ to m **do**
9: $L = L_v$, $I = \overline{X} \setminus L$
10: Perform Cholesky factorization $R_{I,I} = S_L^t S_L$
11: Compute $\log\det(R_{I,I}) = 2\log\det(S_L)$
12: Compute $D_{L,L}$ (eq. (18))
13: **for** $l \in L$ **do**
14: Perform Cholesky factorization of B_ℓ (eq. (16)
15: Compute $\log\det(R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}})$ (eq. (14))
16: Compute $\overline{\delta}(\ell)$ (eq. (11))
17: **end for**
18: **end for**
19: $l^* = \arg\max\, \overline{\delta}(\ell)$, $X = X \cup \{\ell^*\}$
20: **end for**

we address in this subsection the determination of the optimal number of blocks used in Algorithm Blocks to build a design of size n from M candidate points, $m^\star(M, n)$. Details of the analysis are relegated to Appendix A.

The algorithmic complexity of Algorithm Blocks is dominated by the Cholesky factorizations, whose cost is equal to $O(p^3)$ for a $p \times p$ matrix. As shown in Appendix A the total (for all iterations) cost of instructions 10 and 12 increases with m; see the expression for $c_{10} + c_{12}$ in the Appendix, while the cost of instruction 14, corresponding to c_{14} in the Appendix, decreases. This results, for large values of M, in a total cost of order $O[(M-n)^4/m^3]$ to build a design of size n.

In Appendix A, see (20), we derive upper and lower bounds on the optimal value $m^*(M, n)$, respectively $m^+(M, n)$ and $m^-(M, n)$. It is found that $m^*(M, n) \approx m^-(M, n) = [3(M-n)]^{\frac{1}{4}}$ when $M \to \infty$ while keeping n fixed.

This indicates that block size should vary as the design grows during sequential construction of the design. However, since in practical applications $n \ll M$, we consider a fixed value of m for all iterations, and propose to use $m = \widehat{m}(M)$, with

$$\widehat{m}(M) = m^-(M, 0) = \lceil (3M)^{\frac{1}{4}} \rceil. \tag{19}$$

Using the lower bound $m = m^-(M, n)$ leads to a total cost of $O[(M-n)^{3.25}]$ for Algorithm Blocks, showing a clear improvement from the worst case complexity of $O(nM^4)$ of Algorithm Lazy.

We want to stress the generality of the analysis above. First, note that it is independent of the correlation kernel $r(x, y)$ of the Gaussian process used to build the correlation matrix $R_{\mathbf{D}_M,\mathbf{D}_M}$, which adds to the complexity above a constant term, of order $M \times M$. Moreover, it is also insensitive to the exact partition $\{L_v\}_{v=1,\ldots,m}$ of the set of candidate points $\overline{X}$ that is used. Finally, it is as well independent of the geometry of the finite set of candidate design points $\mathbf{D}_M$ that covers the domain D.

The same is not true for Algorithm Lazy. Since the construction of list $\mathcal{M}$ in each of its iterations depends both on the correlation structure $r(x, y)$ of the process and on the order by which points are tested, its complexity in practical problems is expected to depend on the chosen Gaussian process model as well on the manner in which $\mathbf{D}_M$ is embedded in D, that is, on the geometry of $\mathbf{D}_M$. The numerical study presented in section 5 confirms this.

4.3 Numerical illustration

We present in this subsection numerical studies that confirm our analytical model of the variation algorithmic complexity of Algorithm Blocks with m as well as the validity of $\widehat{m}(M)$ given by (19) as an approximation for the optimal partition size.

Figure 1 plots the evolution of the measured computational time taken by Algorithm Blocks for the generation of N-point designs, with N varying from 20 to 120 by steps of 20. The candidate design points are the first $M = 500$ points of the 2-dimensional Sobol' sequence, see Sobol and Levitan (1976), and the kernel of the Gaussian process is a Matérn 5/2,

$$K_{5/2,\theta}(x, x') = \left[1 + \sqrt{5}\,\theta\,\|x - x'\| + 5\,\theta^2\,\|x - x'\|^2/3\right] \exp(-\sqrt{5}\,\theta\,\|x - x'\|)\,,$$

see for instance Stein (1999), with correlation range $\theta = 0.08$. Similar results were obtained for other correlation kernels (exponential and Gauss). The Figure confirms our analysis, showing that the optimal value of m is largely independent of N, all the curves exhibiting a minimum at about the same value. It also confirms the near optimality of the value $\widehat{m}(M)$ predicted by equation (19): $\widehat{m}(M) \simeq 7$, indicated by the red vertical line, which is only slightly larger than the minima observed. Note that since the increase of computational complexity when $m \gtrapprox m^\star$ is slow, this slight overestimation of the optimal number of blocks m^* is preferable to its underestimation, inducing only a small penalty in the computational complexity.

Figure 2 shows the impact of the size of $\mathbf{D}_M$ on the number of blocks $m^\star$ that leads to a minimal complexity of Algorithm Blocks. For several values of m we measured the time required to select a design of a fixed size ($N = 100$) for $M \in \{500, 1\,000, 2\,000\}$. Since algorithm complexity quickly grows with M, to facilitate the analysis the plots show normalised versions $\hat{t} \in [0, 1]$ of the measured running times. Each curve in Figure 2 is thus affected by a distinct normalisation and for that reason the scale of the vertical axis is omitted. The plot on the right shows a zoom around $m \simeq m^*$ which confirms the increase of the optimal block size m^* with M predicted by equation (19) (indicated in the legend of the figure). The Figure also shows that the quality of the approximation increases with M, being better for $M = 2\,000$ than for $M = 500$.

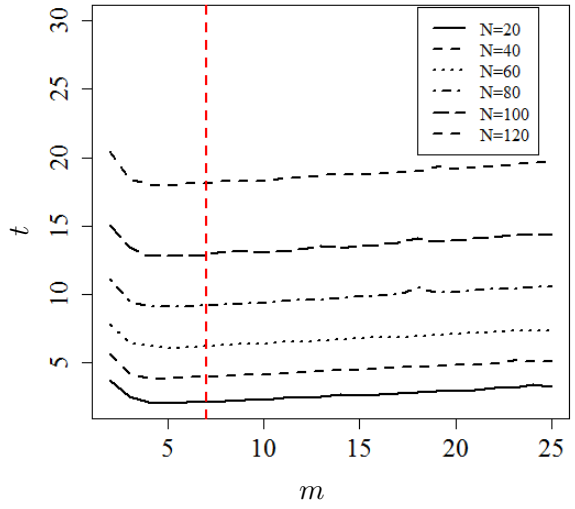

Figure 1: Computational time t (in seconds) for the generation of N-point designs as a function of m; $N = 20, 40, 60, 80, 100, 120$, $d = 2$, $M = 500$. Matérn Gaussian process kernel.

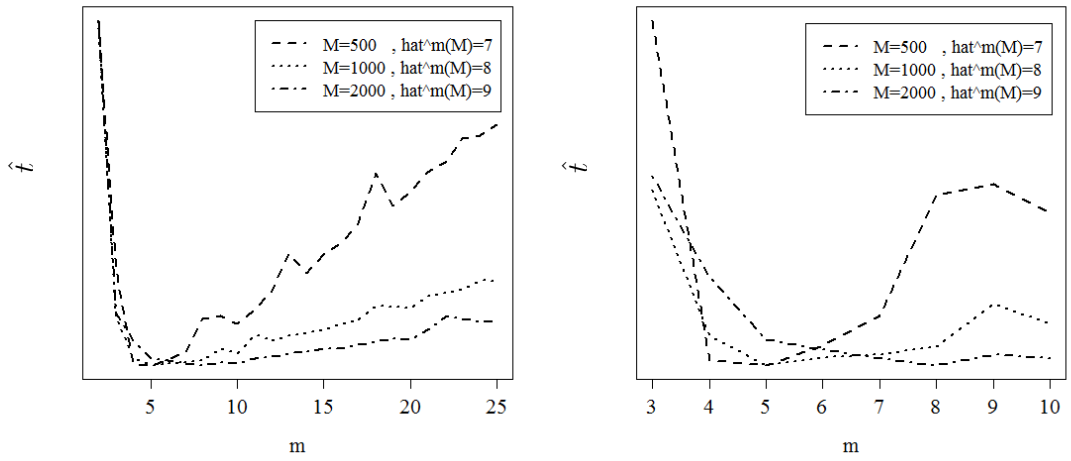

Figure 2: (normalized) computational time $\hat{t}$ for the generation of a N-point design ($N = 100$) as a function of number of blocks m for $M = 500, 1\,000, 2\,000$; $d = 2$; the right panel is a zoom of the left one around m^*. Matérn Gaussian process kernel.

5. Combining submodularity and matrix partitioning

In this section we present a third algorithm, that combines the lazy principle of Algorithm Lazy with the organisation of the numerical computations on which Algorithm Blocks is based. A thorough numerical study of the complexity of the three algorithms is presented enabling comparison of their efficiency.

5.1 Approach

Algorithm Lazy-Blocks, see page 23, draws on the ideas presented in both sections 3.3 and 4.1, combining the parsimonious evaluation of candidate points of Algorithm Lazy with the computation sharing based on partition of the correlation matrix as done by Algorithm Blocks.

The "Initialisation" section (lines 1–10) uses Algorithm Blocks to compute the individual increments of each $x \in \mathbf{D}_M$ with respect to the empty set. The same block structure is used to share the partial computations amongst the elements of each element of the m-ary partition build in the first instruction. The point with largest value of $\overline{\delta}(x)$ is then chosen as the first point (line 13). From that singleton solution the algorithm increases the size of the design X by adding one point at a time. The computational organisation of Algorithm Blocks is then combined with lazy-greedy by working in each iteration with a new partition of the remaining candidate points, build in the first instruction of the greedy cycle (line 17). As in Algorithm Blocks, $\log\det(R_{X\cup\{x\},X\cup\{x\}})$, the first term in the expression for $\overline{\delta}(x)$, which uses the Cholesky factorisation of the design matrix, is then computed for all the remaining candidate points (lines 18–21).

The algorithm proceeds to the calculation of the second term in equation (11), by considering in turn each block L_v of the partition. For each block, Algorithm Lazy is applied to the corresponding elements. Two lists, Lists $\mathcal{L}^+$ and $\mathcal{L}^-$, are instrumental in the implementation of the algorithm: $\mathcal{L}^-$ contains the set points (of block L_v) that remain potential best choices given the updated values of $\overline{\delta}$ and $\mathcal{L}^+$ gathers the set of points (in all blocks) that either have been updated or that are worse than the best point found. At each iteration, the remaining candidate points that are worse than the best updated point (the list $\mathcal{M}$, updated in line 42) are added to $\mathcal{L}^+$ (line 43). Note that the algorithm works "across blocks": the increments of elements that are placed in $\mathcal{L}^+$ when screening a different block than the one they belong to will not be considered when their own block is processed. Whenever $\overline{\delta}(x)$ is actually computed (line 40), re-use of previous matrix computations is done as for Algorithm Blocks.

It is important to note that, as we mentioned before, the set of elements whose increment is actually computed in Algorithm Lazy depends on the relative strength of correlation with the points already in the design, and thus its complexity is potentially affected by the choice of $r(x, y)$ and by the geometry of the set of candidate points. Moreover, it is also affected by the order by which the candidate points are selected at each iteration: orderings that lead to lists $\mathcal{M}$ which have large sizes lead to larger computational gains. Since all elements in each member of the partition are screened consecutively, the partition $\{L_v\}_{v=1,\ldots,m}$ used influences the final complexity of Algorithm Lazy-Blocks. In our current implementation, a new partition is built at each iteration, as the uniform partition of the set of candidate points sorted by decreasing order of $\overline{\delta}$.

Table 1: Variances of running times (20 runs)

$(a, C, \mathbf{D}_M)$	$N = 50$	$N = 100$	$N = 150$	$N = 200$
(Lazy, Matérn, grid)	0.050	0.052	0.09	0.073
(Lazy, Matérn, Sobol)	0.015	0.033	0.0168	0.029
(Blocks, Matérn, grid)	0.0009	0.0017	0.0172	0.0499
(Blocks, Matérn, Sobol)	0.0013	0.0022	0.0131	0.0300
(Lazy-Blocks, Matérn, grid)	0.0057	0.0188	0.0248	0.0918
(Lazy-Blocks, Matérn, Sobol)	0.0123	0.0167	0.0213	0.0503
Lazy, Exponential, grid)	0.0213	0.0960	0.0494	0.0370
(Lazy, Exponential, Sobol)	0.0146	0.0331	0.0168	0.0290
(Blocks, Exponential, grid)	0.0218	0.0224	0.0524	0.0845
(Blocks, Exponential, Sobol)	0.0123	0.0167	0.0213	0.0503
(Lazy-Blocks, Exponential, grid)	0.0013	0.0014	0.0114	0.0296
(Lazy-Blocks, Exponential, Sobol)	0.0013	0.0022	0.0131	0.0300

5.2 Numerical comparison of the three algorithms

We present in this section a numerical study of the complexity of algorithms Lazy, Blocks and Lazy-Blocks, assessed through the running times of 20 executions of each algorithm, $T^a_{d,C,g}(M, N)$, $a \in \{\text{Lazy}, \text{Blocks}, \text{Lazy} - \text{Blocks}\}$. In the notation, $d \in \{1, 5\}$ indicates the dimension of the input space of f, $C \in \{\text{Matérn}, \text{Exponential}\}$ indicates the correlation function of the Gaussian process model, $g \in \{\text{grid}, \text{Sobol}\}$ indicates the geometry of the set of candidate points $\mathbf{D}_M$ (either a uniform grid or points of the Sobol sequence), M is the size of $\mathbf{D}_M$ and N the final size of the design.

The variances of the running times observed in the 20 runs of each algorithm are shown in Table 1. In our experiments, the standard deviation observed in the 20 runs is always negligible, never exceeding 0.3% of the running time (see the left plots in Figure 3 for the corresponding mean values). For that reason, we concentrate below in the comparison of average running times.

Below, we study how the numerical complexity $T^a_{d,C,g}(M, N)$ of the three algorithms is affected by a number of parameters and implementation choices:

1. Algorithm choice, that is, different values of $a \in \{\text{Lazy, Blocks, Lazy-Blocks}\}$;
2. The geometry g of the set of candidate points $\mathbf{D}_M$;
3. The size M of $\mathbf{D}_M$;
4. The dimension d of the input space;
5. The correlation kernel C of the Gaussian process.

We start by assessing issue 1. The left plots of Figure 3 show the average running times $T^a_{2,C,\mathbf{D}_M}(529, N)$ for algorithms Lazy, Blocks and Lazy-Blocks for several choices of C, $\mathbf{D}_M$ and N. Input space has dimension $d = 2$ and $M = 529$ for all cases. In the top plots a

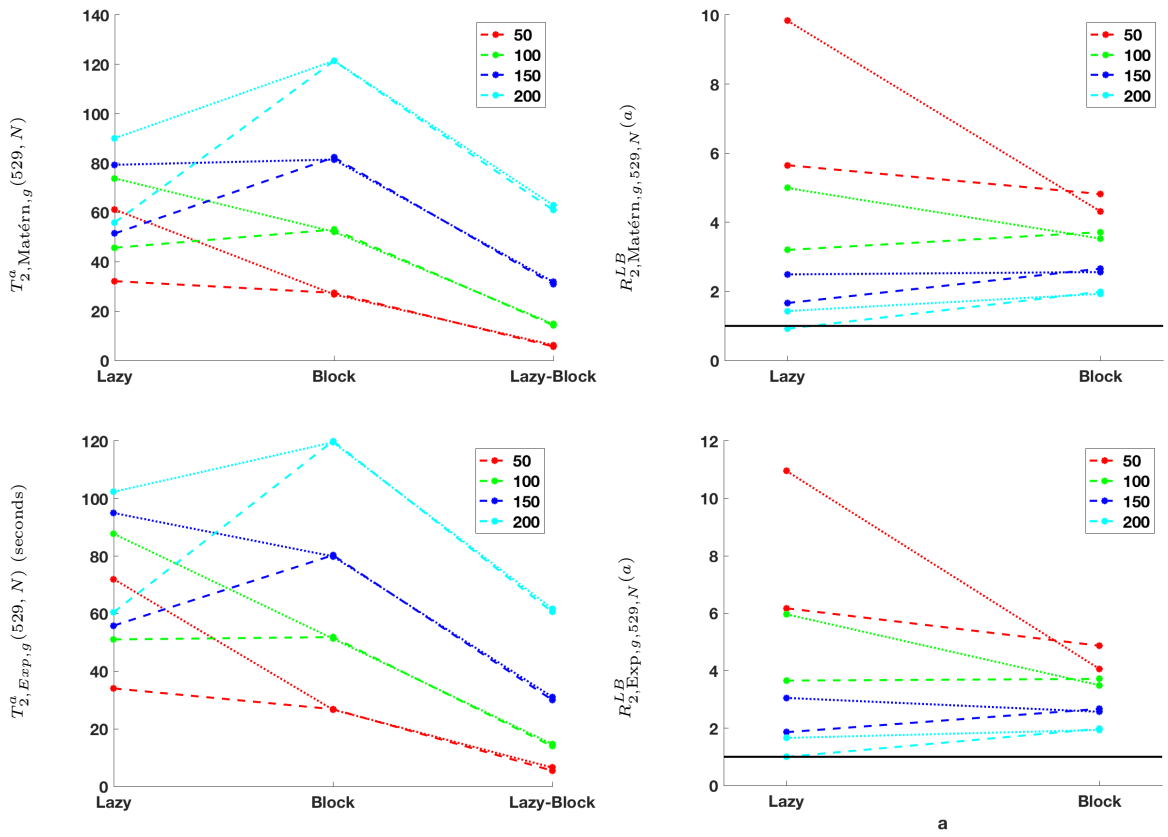

Figure 3: Comparison of running times of algorithms Lazy, Blocks and Lazy-Blocks. For all cases, $d = 2$, $M = 529$ and $N \in \{50, 100, 150, 200\}$, see legend for colour code. Dotted lines correspond to $g =$ grid, dashed lines to $g =$ Sobol. Top: Matern kernel; bottom: exponential kernel. Left: $T^a_{2,C,g}(529, N)$, right: $T^a_{2,C,g}(529, N)/T^{Lazy-Blocks}_{2,C,g}(529, N)$, $a \in \{\text{Lazy}, \text{Block}\}$.

Gaussian process model with a Matérn kernel is used, while those in bottom assume an exponential kernel. The lines link values that differ only on the algorithm chosen. Dotted lines indicate that $g =$ grid, and for dashed lines $g =$ Sobol. The color of the lines indicates the value of $N \in \{50, 100, 150, 200\}$ as indicated in the legend of the figure.

To clarify the gains achieved by the combination of the Lazy and Blocks approaches, the plots in the right of Figure 3 show the corresponding ratios

$$R^{LB}_{d,C,g,M,N}(a) = \frac{T^a_{d,C,g}(M, N)}{T^{Lazy-Blocks}_{d,C,g}(M, N)}, \qquad a \in \{\text{Lazy}, \text{Block}\}, g \in \{\text{grid, Sobol}\} \ .$$

We can see that except in one case (when $g =$Sobol and $N = 200$) all values are larger than 1 (black horizontal line) indicating that Lazy-Blocks has the smallest running time. We can further notice, see plots in the right, that the computational gain offered by Lazy-Blocks

decreases monotonically with N: while a gain greater than 5 is obtained for $N = 50$ (red lines) this relative advantage disappears when $N = 200$, the Lazy Algorithm having then for g =Sobol a running time comparable to Lazy-Blocks, depending on the correlation kernel of the Gaussian process model. Note that for $N = 200$ and $M = 529$ the hypothesis behind our approximation of $m^\star$—that the size of the partitions is very large—is violated, and thus our implementation of Lazy-Blocks may not be optimal. For moderate design sizes the gain of Lazy-Blocks with respect to Lazy is considerable, ranging from around 10 (for the regular grid domain) for $N = 50$ to more than 2 for $N = 150$.

Figure 3 also allows the comparison of the numerical complexity of algorithms Lazy and Block, showing that for the smallest value of N (the exact range of design sizes depending on the intrinsic geometry of $\mathbf{D}_M$) Algorithm Blocks is more efficient than Algorithm Lazy. The running time of Algorithm Blocks is the one that is the most sensitive to the increase of N, and for large values of N (cyan lines) its efficiency is the worst of all three algorithms. Mutualisation of the computations seems thus to be interesting mainly as a way to improve on the Lazy algorithm, offering an advantage over Lazy only for small values of N.

Figure 3 also answers question 2 through the comparison of the dotted (g = grid) and dashed (g = Sobol) lines of the same color. As anticipated, it shows that only the Lazy algorithm is clearly affected by the choice of the geometry of the set of candidate points, the dashed and dotted lines being almost coincident for the Algorithms Blocks and Lazy-Blocks. The numerical complexity of the Lazy algorithm is smaller when $\mathbf{D}_M$ are the points of the Sobol sequence than when a uniform grid is used (the dotted lines are always above the dashed lines). That Algorithm Blocks should not be affected by the particular choice of candidate points was expected, since it updates the increments $\overline{\delta}(x)$ for all points in each iteration. However, the little variation observed for the Lazy-Blocks algorithm is somewhat unexpected.

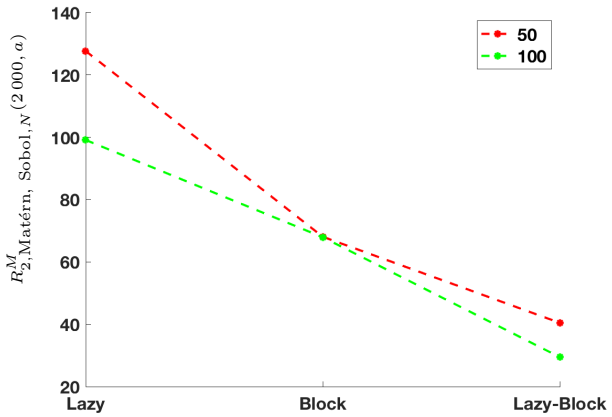

Figure 4: Influence of M. Ratios $R^M_{d,C,g,M,N}(a)$, $a \in \{\text{Lazy, Block, Lazy-Blocks}\}$, $N = 50, 100$.

Figure 4 addresses issue 3, the effect of M on the complexity of the three algorithms. It plots the ratio

$$R^{M}_{d,C,g,N}(M,a) = \frac{T^{a}_{d,C,g}(M,N)}{T^{a}_{d,C,g}(529,N)}, \qquad a \in \{\text{Lazy}, \text{Block}, \text{Lazy-Blocks}\}, N \in \{50, 100\} \ ,$$

for $M = 2\,000$ and moderate design sizes of $N = 50$ and $N = 100$ (red and green lines, respectively). A Matérn kernel is considered, $d = 2$ and $g =$Sobol. We can see that Lazy is penalised the most by the increase of the size of candidate points from 529 to 2 000: its complexity is multiplied by 130 for the largest design size and by 100 for $N = 50$, when the size M of the set of candidate points is increased only by 3.7. As we should expect, the penalty of Algorithm Blocks does not depend on design size N, being close to 70 for both $N = 50$ and $N = 100$. The plots show an increase in numerical complexity of Algorithm Lazy-Blocks of 30 and 40, for $N = 100$ and $N = 50$, respectively. We can thus expect, in the light of these results, that the range of designs sizes for which Algorithm Blocks is more efficient than Lazy, which was approximately $N \leq 100$ in Figure 3, can be considerably larger for larger values of M.

Figure 5 compares the complexity of the three algorithms for two different dimensions of the input space, $d = 2$ and $d = 5$, plotting the ratio

$$R^{d}_{C,g,M,N}(d,a) = \frac{T^{a}_{d,C,g}(M,N)}{T^{a}_{2,C,g}(M,N)}, \qquad a \in \{\text{Lazy}, \text{Block}, \text{Lazy-Blocks}\},$$

for the construction of designs of size $N = 50$ and $N = 100$, addressing thus question 4. In both cases $\mathbf{D}_M$ corresponds to the first $M = 2\,000$ points of the Sobol sequence. Following (19), Algorithms Blocks and Lazy-Blocks use $\widehat{m} = 9$. As expected, the computational time for Algorithm Blocks is not impacted by the dimension of the input space if M is held constant. In contrast Algorithm Lazy slows down as dimension increases, the degradation being more pronounced for the exponential (right plot) than for the Matérn correlation kernel (left plot). The complexity penalty grows with design size N, being equal to 1.66 for $N = 50$ and slightly larger than 2 for $N = 100$ (Matérn correlation). Algorithms Blocks and Lazy-Blocks are virtually insensitive to the increase of d, their ratios (see plot in the left of Figure 5) being close to 1 and virtually independent of the design size N.

Figure 6 plots the ratio of the running times of Algorithms Lazy and Blocks to the running time of Lazy-Blocks for $d = 2$ and for $d = 5$ (all other parameters are held constant: $N = 50, M = 2\,000$, $g =$Sobol, $C =$Matérn). It shows that the gain of Lazy-Blocks relative to Lazy is the largest when $d = 5$, when it is nearly equal to 30, almost twice as the gain for $d = 2$.

We finally address issue 5, the influence of the choice of the correlation kernel on the complexity of the design algorithms. Comparison the top and bottom plots in Figure 3 shows that the chosen correlation kernel has little influence in the relative behaviour of the algorithms: the geometry of the lines varies little between the top and bottom plots. However, comparison of the actual running times of this Figure, as well as analysis of the red lines in the left and right plots of Figure 5, indicate that the Exponential kernel leads, for Algorithm Lazy, to consistently larger running times than the Matérn kernel. Figure 7 plots combinations of Matérn/Exponential and Grid/Sobol for Algorithm Lazy, sowing

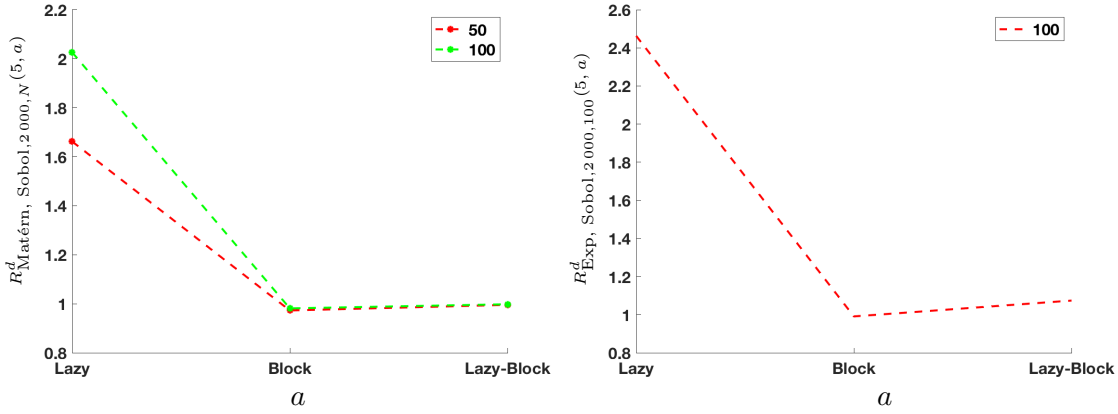

Figure 5: Influence of d on running time. Ratios $R^d_{C,\text{Sobol},2\,000,N}(5,a)$, for algorithms Lazy, Blocks and Lazy-Blocks. Left: $C =$ Matern, $N = 50, 100$; right: $C =$ Exponential, 100.

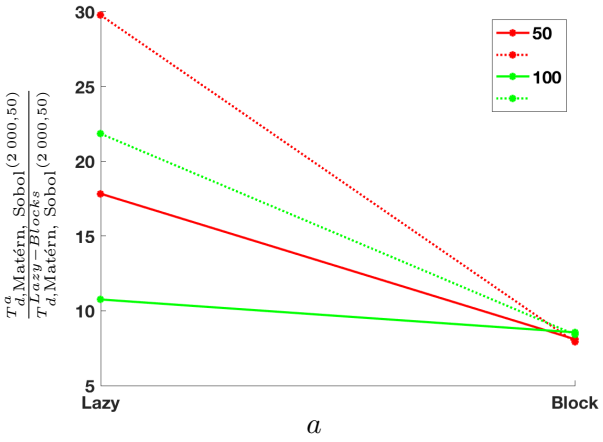

Figure 6: Influence of d on relative efficiency of Lazy-Blocks. Solid lines: $d = 2$, dotted lines: $d = 5$. Red: $N = 50$, green: $N = 100$.

that changing from the Matérn to the exponential correlation kernel induces a penalty approximately independent of N, which is larger for the grid geometry (red points) than when $\mathbf{D}_M$ is the Sobol set (green points).

We summarise below the major conclusions

1. The complexity of both Blocks and Lazy-Blocks depends only on the size of the set of candidate design points, being insensitive to the choice of correlation kernel, the geometry of the set of candidate design points, and the dimension of the input space. On the contrary, Algorithm Lazy is sensitive to all parameters and choices, being

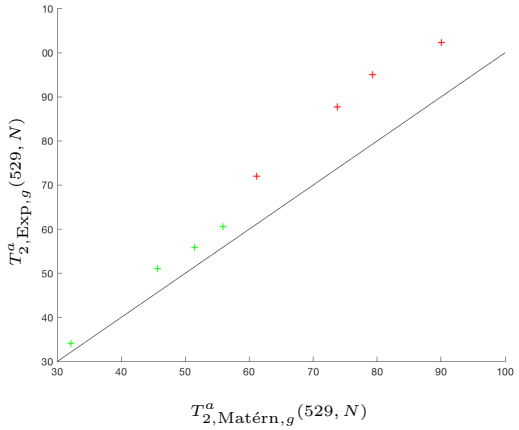

Figure 7: Influence of correlation kernel on the complexity of Algorithm Lazy. Red: g =grid; green: g =Sobol. Symbols of the same colour differ in the value of $N \in \{50, 100, 150, 200\}$.

larger for the exponential kernel than for the Matérn kernel, and for the grid topology than for Sobol sequence.

2. The complexity of Algorithm Lazy-Blocks is in general the smallest of all three algorithms, the gain in computational time with respect to Lazy being larger for small designs (N) and larger dimensions of the input space (d).

6. Conclusion

An efficient implementation of a greedy algorithm for the construction of designs with maximal Mutual Information (Algorithms Blocks and Lazy-Blocks) has been proposed. The algorithm is based on a block partition of the correlation matrix which allows re-use of the result of matrix operations of large numerical complexity, like Cholesky-based matrix inversions. A quasi-optimal rule for the number of blocks in the partition is presented, based on a detailed analysis of the complexity of the algorithm.

Combining this organisation of the numerical computations involved in the evaluation of the MI criterion with the strategy behind Algorithm Lazy originally presented in Krause et al. (2008), which exploits the sub-modularity of the MI criterion to avoid useless criterion computations, the paper proposes the new algorithm Lazy-Blocks, which may offer important reductions in the computational load involved in the sequential construction of an experimental design, in particular when the dimension of the input space is large and the design is a small subset of the complete set of candidate points, as it is the case in most situations of practical interest.

Contrary to the Lazy algorithm proposed in Krause et al. (2008) the complexity of the proposed algorithm is insensitive to the choice of correlation kernel, to the intrinsic geometry of the set of candidate points as well as to the dimension of the input space, depending only on the size of the solution set, which is determined by the size of the set of candidate points

and the size of the design. Algorithm Lazy-Blocks is thus an efficient and robust alternative to previously proposed algorithms for the sequential construction of MI-optimal designs.

Acknowledgments

The authors acknowledge funding from the OQUAIDO chair and ANR PEPITO for the work presented, and thank Bertrand Iooss for initially motivating them to this problem, drawing their attention to the importance of the sequential construction of space-filling design in industrial applications. The two last authors acknowledge partial funding from ANR project INDEX, contract ANR-18-CE91-0007.

References

F. Bach. Learning with submodular functions: a convex optimization perspective. *arXiv preprint arXiv:1111.6453*, 2011.

W.F. Caselton and J.V. Zidek. Optimal monitoring network designs. *Statistics and Probability Letters*, 2(4):223–227, 1984.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons" 1991.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques. Proc. 8th IFIP Conference, Wurzburg, 1977 (Part 2)*, pages 234–243. Springer, New-York, 1978.

B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1812–1818. AAAI Press, 2015.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

I.M. Sobol and Yu.L. Levitan. The production of points uniformly distributed in a multidimensional cube. *Preprint IPM Akad. Nauk SSSR*, 40(3), 1976.

M.L. Stein. *Interpolation of Spatial Data. Some Theory for Kriging*. Springer, Heidelberg, 1999.

Appendix A. Optimal size of partitions

We estimate the cost in Algorithm Blocks of adding one point chosen among t points ($t = M - n$ in the algorithm, with n the current design size and M the total number of candidate design points). Instructions 3 and 5 have negligible cost compared to that of other instructions; moreover, their cost does not depend on m. Therefore, we limit ourselves to evaluating complexity of instructions 10, 12 and 14.

Hypotheses We make the approximation that the partition is uniform: $\mathrm{card}(L_v) = t/m$ for all v in $\{1, ..., m\}$.

Lemma 1 *Approximate numerical complexity of elementary matrice operations*
The cost of the Cholesky decomposition of a $p \times p$ matrix is $p^3/3 + p^2/2 + p/2$ and the cost of the solution to a triangular system of size p is p^2.

1. The cost of the Cholesky decomposition of matrices of size $\mathrm{card}(J \setminus L_v) = t(1 - 1/m)$ is
$$\frac{t^3}{3}\left(1 - \frac{1}{m}\right)^3 + \frac{t^2}{2}\left(1 - \frac{1}{m}\right)^2 + \frac{t}{2}\left(1 - \frac{1}{m}\right),$$
and thus the cost of m passes through instruction 10 is
$$c_{10} = t^3\frac{(m-1)^3}{3m^2} + t^2\frac{(m-1)^2}{2m} + t\frac{(m-1)}{2}.$$

2. The cost of the solution of $\mathrm{card}(L_v) = t/m$ triangular systems of size $\mathrm{card}(J \setminus L_v)$ required in line 12 to compute C_L, see eq. (18), is
$$\frac{t}{m} \cdot t^2\left(1 - \frac{1}{m}\right)^2 = \frac{t^3}{m}\left(\frac{m-1}{m}\right)^2.$$
To this cost we must add the cost of the product of a $\mathrm{card}(L_v) \times \mathrm{card}(J \setminus L_v) = t/m \times t(1 - 1/m)$ matrix by its transpose involved in the evaluation of $D_{L,L}$ in line 12, which is
$$2\left(\frac{t}{m}\right)^2 \cdot t\left(1 - \frac{1}{m}\right) = 2\frac{t^3}{m^3}(m-1).$$
Thus the cost of m passes through instruction 12 is
$$c_{12} = t^3\left(\frac{m-1}{m}\right)^2 + t^3\frac{2(m-1)}{m^2}.$$

3. Line 14, see equation (16), requires the computation of the difference between square matrices of size $\mathrm{card}(L_v) - 1 = t/m - 1$ whose cost is
$$\left(\frac{t}{m} - 1\right)^2,$$
and since it is executed t times the total cost of this differences is
$$c_{14_{diff}} = \frac{t^3}{m^2} - \frac{2t^2}{m} + t.$$

The cost of each Cholesky decomposition of a matrix of size $\mathrm{card}(L_v)-1$ in line 14 is

$$\frac{1}{3}\left(\frac{t}{m}-1\right)^3+\frac{1}{2}\left(\frac{t}{m}-1\right)^2+\frac{1}{2}\left(\frac{t}{m}-1\right),$$

and thus, for the t passes through instruction 14, a total cost of

$$\begin{aligned}c_{14_{ch}} =& t\left(\frac{t^3}{3m^3}-\frac{t^2}{m^2}+\frac{t}{m}-\frac{1}{3}+\frac{t^2}{2m^2}-\frac{t}{m}+\frac{1}{2}+\frac{t}{2m}-\frac{1}{2}\right)\\ =&\frac{t^4}{3m^3}-\frac{t^3}{2m^2}+\frac{t^2}{2m}-\frac{t}{3}.\end{aligned}$$

The cost of instructions 10 and 12 is

$$\begin{aligned}c_{10}+c_{12} &= (m-1)\left(\left[\frac{(m-1)^2}{3m^2}+\frac{(m-1)}{m^2}+\frac{2}{m^2}\right]t^3+\frac{m-1}{2m}t^2+\frac{t}{2}\right)\\ &= (m-1)\left(\frac{m^2+m+4}{3m^2}t^3+\frac{(m-1)}{2m}t^2+\frac{t}{2}\right),\end{aligned}$$

which increases with m. The cost of instruction 14 is

$$c_{14}=c_{14_{diff}}+c_{14_{ch}}=\frac{t^4}{3m^3}+\frac{t^3}{2m^2}-\frac{3t^2}{2m}+\frac{2t}{3},$$

which decreases with m.

The total cost of each greedy iterations is thus

$$\begin{aligned}c(m) =& c_{10}+c_{12}+c_{14}\\ =&\frac{t^4}{3m^3}+\left(\frac{2m^3+6m-5}{6m^2}\right)t^3+\left(\frac{m^2-2m-2}{2m}\right)t^2+\left(\frac{m}{2}+\frac{1}{6}\right)t.\end{aligned}$$

For fixed m, Algorithm Blocks has $O(t^4)$ complexity when $t\to+\infty$. Writing $c(m)$ as a polynomial in m,

$$c(m)=\left(\frac{t^3}{3}+\frac{t^2}{2}+\frac{t}{2}\right)m+\frac{t}{6}-t^2+\left(t^3-t^2\right)\frac{1}{m}-\frac{5t^3}{6m^2}+\frac{t^4}{3m^3}.$$

For large t, $c(m)$ can be approximated by

$$\widehat{c}(m)=\frac{t^3m}{3}-t^2+\frac{t^3}{m}-\frac{5t^3}{6m^2}+\frac{t^4}{3m^3}.$$

The function $\widehat{c}(\cdot): m\to\widehat{c}(m)$ is continuous on $[1,+\infty[$, with $\lim_{m\to+\infty}\widehat{c}(m)=+\infty$; therefore, $\widehat{c}$ has at least one global minimum on $[1,+\infty[$. Its first two derivatives are given by

$$\begin{aligned}\widehat{c}'(m) &= \frac{\mathrm{d}\widehat{c}(m)}{\mathrm{d}m}=\frac{t^3}{m^4}\left(\frac{m^4}{3}-m^2+\frac{5m}{3}-t\right),\\ \widehat{c}''(m) &= \frac{\mathrm{d}^2\widehat{c}(m)}{\mathrm{d}m^2}=\frac{t^3}{m^5}\left(2m^2-5m+4t\right).\end{aligned}$$

Since $2m^2 - 5m + 4t$ does not admit any real root on $[1, +\infty[$ and $\widehat{c}''(1) > 0$, $\widehat{c}$ is strictly convex on $[1, +\infty[$ and has a unique minimum, which we denote m^*.

Define $m^- = (3t)^{\frac{1}{4}}$, root of $m^4/3 - t = 0$. For this value of m, the first derivative of the (approximate) cost is negative: $\widehat{c}'(m^-) = [t^3/(m^-)^2]\,[(5/(3m^-) - 1]) < 0$. Let now

$$m^+ = \left(\frac{3}{2}\left(1 + \sqrt{1 + \frac{4}{3}t}\right)\right)^{\frac{1}{2}} ,$$

denote a root of $m^4/3 - m^2 - t = 0$, for which it can be seen that the first derivative of $c(m)$ is positive: $\widehat{c}'(m^+) = 5t^3/[3(m^+)^3] > 0$. The value $m^\star$ at which $\hat{c}(m)$ is minimum must then lie between these two values:

$$m^- \leqslant m^* \leqslant m^+ . \tag{20}$$

Finally, remark that if we let $t \to +\infty$ then the optimal value $m^* \to m^-$ and $c(m^-) \simeq \hat{c}(m^-) = O(t^{3.25})$.

Algorithm C (Algorithm Lazy-Blocks)

{**I**nitialization}
Partition $\mathbf{D}_M$: $X = \emptyset\,, \overline{X} = \bigcup_{v=1}^{m} L_v$
for $v = 1, \ldots, m$ **do**
  $L = L_v\ ,\ I = \overline{X} \setminus L$
  Cholesky factorization $R_{I,I} = S_L^t S_L$; compute $\log\det(R_{I,I}) = 2\log\det(S_L)$
  Compute $B_{L,L}$ (equation (18))
  for $x \in L$ **do**
    Cholesky factorization of B_x; compute $\log\det(R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}})$ (eqs. (17,14))
    Compute $\overline{\delta}(x)$ (equation (11))
  end for
end for

{Select first point}
$x_1 = \arg\max_x\ \overline{\delta}(x)$; set $X = \{x_1\}$;

{Incrementally build the design}
for $n = 2$ to N **do**
  Partition $\overline{X} = \bigcup_{v=1}^{m} L_v$, set $\mathcal{M} = \emptyset$,
  Cholesky factorization of current design matrix $R_{X,X} = S^t S$
  for all $\ell \in \overline{X}$ **do**
    Compute $\log\det(R_{X\cup\{x\},X\cup\{x\}})$ (equation (12))
  end for

  {Treat each block of the partition separately}
  $\mathcal{L}^+ = \emptyset$
  for $v = 1, \ldots, m$ **do**
    $L = L_v\ ,\ I = \overline{X} \setminus L$
    Perform Cholesky factorization $R_{I,I} = S_L^t S_L$
    Compute $\log\det(R_{I,I}) = 2\log\det(S_L)$
    Compute $B_{L,L}$ (equation (18))
    Chose a $x \in L_v$;
    Cholesky factorization of B_x; compute $\log\det(R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}})$ (eqs. (17,14))
    Compute $\overline{\delta}(x)$; (equation (11))
    $\mathcal{L}^+ = \mathcal{L}^+ \cup \{x\}$; $\mathcal{L}^- = L_v \setminus \mathcal{L}^+$;

    {Go over block L_v}
    while $\mathcal{L}^- \neq \emptyset$ **do**
      Chose a $x \in \mathcal{L}^-$
      if $\overline{\delta}(x) \geq \max_{x\in\mathcal{L}^+}\ \overline{\delta}(x)$ **then**
        Cholesky factorization of B_x; compute $\log\det(R_{\overline{X}\setminus\{x\},\overline{X}\setminus\{x\}})$ (eqs. (17,14))
        Compute $\overline{\delta}(x)$; (equation (11))
      end if
      $\mathcal{M} = \{x' \in I \cup \mathcal{L}^-\ :\ \overline{\delta}(x') \leq \max_{x\in\mathcal{L}^+}\ \overline{\delta}(x)\}$;
      $\mathcal{L}^+ = \mathcal{L}^+ \cup \mathcal{M}$; $\mathcal{L}^- = \mathcal{L}^- \setminus \mathcal{M}$;
    end while

    {Increment design}
    $x_n = \arg\max_{x\in\mathcal{L}^+}\ \overline{\delta}(x)$;
    $X = X \cup \{x_n\}$;
  end for