A Bayesian optimization approach to find Nash equilibria

Victor Picheny, Mickaël Binois, Abderrahmane Habbal

Noname manuscript No.
(will be inserted by the editor)

A Bayesian optimization approach to find Nash equilibria

Victor Picheny* · **Mickaël Binois*** ·
Abderrahmane Habbal

Received: date / Accepted: date

Abstract Game theory finds nowadays a broad range of applications in engineering and machine learning. However, in a derivative-free, expensive black-box context, very few algorithmic solutions are available to find game equilibria. Here, we propose a novel Gaussian-process based approach for solving games in this context. We follow a classical Bayesian optimization framework, with sequential sampling decisions based on acquisition functions. Two strategies are proposed, based either on the probability of achieving equilibrium or on the Stepwise Uncertainty Reduction paradigm. Practical and numerical aspects are discussed in order to enhance the scalability and reduce computation time. Our approach is evaluated on several synthetic game problems with varying number of players and decision space dimensions. We show that equilibria can be found reliably for a fraction of the cost (in terms of black-box evaluations) compared to classical, derivative-based algorithms. The method is available in the R package `GPGame` available on CRAN at `https://cran.r-project.org/package=GPGame`.

Keywords Game theory · Gaussian processes · Stepwise Uncertainty Reduction

V. Picheny
MIAT, Université de Toulouse, INRA, Castanet-Tolosan, France
Tel.: +33561285551
E-mail: victor.picheny@inra.fr

M. Binois
The University of Chicago Booth School of Business, 5807 S. Woodlawn Ave., Chicago IL, 60637
E-mail: mickael.binois@chicagobooth.edu

A. Habbal
Université Côte d'Azur, Inria, CNRS, LJAD, UMR 7351, Parc Valrose, 06108 Nice, France.
E-mail: habbal@unice.fr

* Both authors contributed equally to this manuscript.

1 Introduction

Game theory arose from the need to model economic behavior, where multiple decision makers (MDM) with antagonistic goals is a natural feature. It was further extended to broader areas, where MDM had however to deal with systems governed by ordinary differential equations, the so-called differential games. See e.g., Gibbons [19] for a nice introduction to the general theory and Isaacs [33] for differential games. Recently, engineering problems with antagonistic design goals and with real or virtual MDM were formulated by some authors within a game-theoretic framework. See e.g., León et al [39] for aerodynamics, Habbal et al [25] for structural topology design, Habbal and Kallel [24] for missing data recovery problems. The study of multi-agent systems or games such as poker under this setting is also quite common in the AI and machine learning communities, see e.g., Johanson and Bowling [35], Lanctot et al [38], Brown et al [7].

Solutions to games are called equilibria. Contrarily to classical optimization, the definition of an equilibrium depends on the game setting (or rules). Within the static with complete information setting, a relevant one is the so-called Nash equilibrium (NE). Shortly speaking, a NE is a fixed-point of iterated many single optimizations (see the exact definition in Section-2 below). Its computation generically carries the well known tricks and pitfalls related to computing a fixed-point, as well as those related to intensive optimizations notably when cost evaluations are expensive, which is the case for most engineering applications. There is an extensive literature related to theoretical analysis of algorithms for computing NE [4, 40, 59], but very little -if any- on black-box models (i.e., non convex utilities) and expensive-to-evaluate ones; to the best of our knowledge, only home-tailored implementations are used. On the other hand, *Bayesian optimization* [BO, 43] is a popular approach to tackle black-box problems. Our aim is to investigate the extension of such approach to the problem of computing game equilibria.

BO relies on Gaussian processes, which are used as emulators (or surrogates) of the black-box model outputs based on a small set of model evaluations. Posterior distributions provided by the Gaussian process are used to design *acquisition functions* that guide sequential search strategies that balance between exploration and exploitation. Such approaches have been applied for instance to multi-objective problems [61], as well as transposed to frameworks other than optimization, such as uncertainty quantification [5] or optimal stopping problems in finance [23].

In this paper, we show that the BO apparatus can be applied to the search of game equilibria, and in particular the classical Nash equilibrium (NE). To this end, we propose two complementary acquisition functions, one based on a greedy search approach and one based on the Stepwise Uncertainty Reduction paradigm [13]. The corresponding algorithms require very few model evaluations to converge to the solution. Our proposal hence broadens the scope of applicability of equilibrium-based methods, as it is designed to tackle derivative-free, non-convex and expensive models, for which a game perspective was previously out of reach.

The rest of the paper is organized as follows. Section 2 reviews the basics of game theory and presents our Gaussian process framework. Section 3 presents our main contribution, with the definition of two acquisition functions, along with com-

putational aspects. Finally, Section 4 demonstrates the capability of our algorithm on three challenging problems.

2 Background

2.1 Games and equilibria

2.1.1 Nash games

We consider primarily the standard (static, under complete information) Nash equilibrium problem [NEP, 19].

Definition 1 A NEP consists of $p \geq 2$ decision makers (i.e., players), where each player $i \in \{1, \ldots, p\}$ tries to solve his optimization problem:

$$(\mathcal{P}_i) \quad \min_{\mathbf{x}_i \in \mathbb{X}_i} y_i(\mathbf{x}), \tag{1}$$

where $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), \ldots, y_p(\mathbf{x})] : \mathbb{X} \subset \mathbb{R}^n \to \mathbb{R}^p$ (with $n \geq p$) denotes a vector of cost functions (a.k.a. pay-off or utility functions), y_i denotes the specific cost function of player i, and the vector $\mathbf{x}$ consists of block components $\mathbf{x}_1, \ldots, \mathbf{x}_p$ ($\mathbf{x} = (\mathbf{x}_j)_{1 \leq j \leq p}$).

Each block $\mathbf{x}_i$ denotes the variables of player i and $\mathbb{X}_i$ its corresponding action space and $\mathbb{X} = \prod_i \mathbb{X}_i$. We shall use the convention $y_i(\mathbf{x}) = y_i(\mathbf{x}_i, \mathbf{x}_{-i})$ when we need to emphasize the role of $\mathbf{x}_i$.

Definition 2 A Nash equilibrium $\mathbf{x}^* \in \mathbb{X}$ is a strategy such that:

$$(NE) \quad \forall i,\ 1 \leq i \leq p, \quad \mathbf{x}_i^* \in \arg\min_{\mathbf{x}_i \in \mathbb{X}_i} y_i(\mathbf{x}_i, \mathbf{x}_{-i}^*). \tag{2}$$

In other words, when all players have chosen to play a NE, then no single player has incentive to move from his $\mathbf{x}_i^*$. Let us however mention by now that, generically, Nash equilibria are not efficient, i.e., do not belong to the underlying set of best compromise solutions, called Pareto front, of the objective vector $(y_i(\mathbf{x}))_{\mathbf{x} \in \mathbb{X}}$.

2.1.2 Random games

We shall also deal with the case where cost functions are uncertain. Such problems belong to a family of random games that are called disturbed games by Harsanyi in [26]. We denote such cost functions $f_i(\mathbf{x}, \boldsymbol{\epsilon}(\xi))$, where $\boldsymbol{\epsilon} = (\epsilon_i) : \Xi \to \mathbb{R}^p$ is a random vector defined over a probability space $(\Xi, \mathcal{F}, \mathbb{P})$. In the following we refer to our setting as random games, but emphasize that we consider *static* Nash games with expectations of randomly perturbed costs.

Definition 3 Assuming risk-neutrality of the players, a random Nash game consists of $p \geq 2$ players, where each player $i \in \{1, \ldots, p\}$ tries to solve

$$(\mathcal{SP}_i) \quad \min_{\mathbf{x}_i \in \mathbb{X}_i} \mathbb{E}[f_i(\mathbf{x}, \boldsymbol{\epsilon}(\xi))]. \tag{3}$$

Definition 4 A random Nash equilibrium $\mathbf{x}^* \in \mathbb{X}$ is a strategy such that:

$$(SNE) \quad \forall i,\ 1 \leq i \leq p, \quad \mathbf{x}_i^* \in \arg\min_{\mathbf{x}_i \in \mathbb{X}_i} \mathbb{E}[f_i(\mathbf{x}_i, \mathbf{x}_{-i}^*, \boldsymbol{\epsilon}(\xi))]. \tag{4}$$

Note that since $\mathbb{E}[f_i(\mathbf{x}, \boldsymbol{\epsilon}(\xi))]$ is not random, one can equivalently set $y_i = \mathbb{E}[f_i(\mathbf{x}, \boldsymbol{\epsilon}(\xi))]$ to formulate the solution as a (NE). The difference lies in the fact that here the cost function y_i cannot be evaluated exactly.

2.1.3 Working hypotheses

In this work, we focus on continuous-strategy non-cooperative Nash games (i.e., with infinite sets $\mathbb{X}_i$) or on large finite games (i.e., with large finite sets $\mathbb{X}_i$).

Our working hypotheses are:

- queries on the cost function (i.e., pointwise evaluation of the y_i's for a given $\mathbf{x}$) result from an expensive process: typically, the y_i's can be the outputs of numerical models;
- the cost functions may have some regularity properties but are possibly strongly not convex (e.g., continuous and multimodal);
- the cost functions evaluations can be corrupted by noise;
- $\mathbb{X}$ is either originally discrete, or a representative discretization of it is available (so that the equilibrium of the corresponding finite game is similar to the one of original problem).

Note that to account for the particular form of NEPs, $\mathbb{X}$ must realize a full-factorial design: $\mathbb{X} = \mathbb{X}_1 \times \ldots \times \mathbb{X}_p$. Given each action space $\mathbb{X}_i = \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{m_i}\}$ of size m_i, $\mathbb{X}$ consists of all the combinations $(\mathbf{x}_i^k, \mathbf{x}_j^l)$ ($1 \leq i \neq j \leq p$, $1 \leq k \leq m_i$, $1 \leq l \leq m_j$), and we have $N := \mathrm{Card}(\mathbb{X}) = \prod_{i=1}^p m_i$.

Let us remark that we do not require for an equilibrium to exist and be unique, the case when there is no or equilibria is discussed in Section 3.2.4.

In the case of noisy evaluations, we consider only here an additive noise corruption:

$$f_i(\mathbf{x}, \boldsymbol{\epsilon}(\xi)) = y_i(\mathbf{x}) + \epsilon_i(\xi), \tag{5}$$

and we assume further that $\boldsymbol{\epsilon}$ has independent Gaussian centered elements: $\epsilon_i \sim \mathcal{N}(0, \tau_i^2)$. Notice that in this case, both problems and equilibria coincide, and can be solved with the same algorithm. Hence, in the following, all calculations are given in the noisy case, while the deterministic case of the standard NEP is recovered by setting $\epsilon_i = 0$ and $\tau_i = 0$.

Furthermore, we consider solely pure-strategy Nash equilibria [as opposed to mixed-strategies equilibria, in which the optimal strategies can be chosen stochastically, see 19, Chapter 1], and as such, we avoid solving the linear programs LP or linear complementarity problems LCP generally used in the dedicated classes of algorithms à la Lemke-Howson [52].

2.1.4 Related work

Let us mention that in the continuous (in the sense of smooth) games setting, there is an extensive literature dedicated to the computation of NE, based on the rich theory of variational analysis, starting with the classical fixed-point algorithms to solve NEPs [59, 4, 40]. When the players share common constraints, Nash equilibria are shown to be Fritz-John (FJ) points [11], which allows the *op. cit.* authors to propose a nonsmooth projection method (NPM) well adapted for the computation of FJ points. From other part, it is well known (and straightforward) that Nash equilibria are in general not classical Karush-Kuhn-Tucker (KKT) points, nevertheless, a notion of KKT condition for generalized Nash equilibria GNEP is developed in Kanzow and Steck [37], which allows the authors to derive an augmented Lagrangian method to compute GNEPs.

Noncooperative stochastic games theory, starting from the seminal paper by Shapley [57], occupies nowadays most of the game theorists, and a vast literature is dedicated to stochastic differential games [14], robust games [45], games on random graphs, or agents learning games [32], among many other branches, and it is definitely out of the scope of the paper to review all aspects of the field. See also the introductory book Neyman and Sorin [44] to the basic -yet deep- concepts of the stochastic games theory.

We also do not consider games with additional specific structures, like cooperative, zero-sum stochastic or deterministic games, or repeated Nash games [41]. For these games, tailored algorithms should be used, among which are the pure exploration statistical learning with Monte Carlo Tree Search [16] or multi-agent reinforcement learning MARL, see e.g., Games [15] and references therein.

We stress here that none of the above-mentioned approaches are designed to tackle expensive black-box problems. They may even prove unusable in this context, either because they could simply not converge or require too many cost function evaluations to do so.

2.2 Bayesian optimization

2.2.1 Gaussian process regression

The idea of replacing an expensive function by a cheap-to-evaluate surrogate is not recent, with initial attempts based on linear regression. Gaussian process (GP) regression, or kriging, extends the versatility and efficiency of surrogate-based methods in many applications, such as in optimization or reliability analysis. Among alternative non-parametric models such as radial basis functions or random forests, see e.g., Wang and Shan [62], Shahriari et al [56] for a discussion, GPs are attractive in particular for their tractability, since they are simply characterized by their mean $m(.)$ and covariance (or kernel) $k(.,.)$ functions, see e.g., Cressie [10], Rasmussen and Williams [51]. In the following, we consider zero-mean processes ($m = 0$) for the sake of conciseness.

Briefly, for a single objective y, conditionally on n noisy observations $\mathbf{f} = (f_1, \ldots, f_n)$, with independent, centered, Gaussian noise, that is, $f_i = y(\mathbf{x}_i) + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, \tau_i^2)$, the predictive distribution of y is another GP, with mean and covariance functions given by:

$$\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1} \mathbf{f}, \tag{6}$$

$$\sigma^2(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}'), \tag{7}$$

where $\mathbf{k}(\mathbf{x}) := (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_n))^\top$ and $\mathbf{K} := (k(\mathbf{x}_i, \mathbf{x}_j) + \tau_i^2 \delta_{i=j})_{1 \leq i,j \leq n}$, δ standing for the Kronecker function. Commonly, $k(.,.)$ belongs to a parametric family of covariance functions such as the Gaussian and Matérn kernels, based on hypotheses about the smoothness of y. Corresponding hyperparameters are often obtained as maximum likelihood estimates, see e.g., Rasmussen and Williams [51] or Roustant et al [53] for the corresponding details.

With several objectives, a statistical emulator for $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), \ldots, y_p(\mathbf{x})]$ is needed. While a joint modeling is possible [see e.g., 2], it is more common practice to treat the y_i's separately. Hence, conditioned on a set of vectorial observations $\{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$, our emulator is a multivariate Gaussian process $\mathbf{Y}$:

$$\mathbf{Y}(.) \sim \mathcal{GP}\left(\boldsymbol{\mu}(.), \boldsymbol{\Sigma}(.,.)\right), \tag{8}$$

with $\boldsymbol{\mu}(.) = [\mu_1(.), \ldots, \mu_p(.)]$, $\boldsymbol{\Sigma} = \mathrm{diag}\left(\sigma_1^2(.,.), \ldots, \sigma_p^2(.,.)\right)$, such that $\{\mu_i(.), \sigma_i^2(.,.)\}$ is the predictive mean and covariance, respectively, of a GP model of the objective y_i. Note that the predictive distribution of an observation is:

$$\mathbf{F}(\mathbf{x}) \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x}, \mathbf{x}) + \mathrm{diag}(\tau_1^2, \ldots, \tau_p^2)\right). \tag{9}$$

GPs are commonly limited to a few thousands of design points, due to the cubic cost needed to invert the covariance matrix. This can be overcome in several ways, by using inducing points [63], or local models [22, 54]; see also [27] for a comparison or [64] for a broader discussion. In addition, here the full-factorial structure of the design space can potentially be exploited. For instance, if evaluated points also have a full-factorial structure, then the covariance matrix can be written as a Kronecker product, reducing drastically the computational cost, see e.g., [49].

2.2.2 Sequential design

Bayesian optimization methods are usually outlined as follows: a first set of observations $\{\mathbf{X}_{n_0}, \mathbf{f}_{n_0}\}$ is generated using a space-filling design to obtain a first predictive distribution of $\mathbf{Y}(.)$. Then, observations are performed sequentially by maximizing a so-called *acquisition function* (or infill criterion) $J(\mathbf{x})$, that represents the potential usefulness of a given input $\mathbf{x}$. That is, at step $n \geq n_0$,

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x} \in \mathbb{X}} J(\mathbf{x}). \tag{10}$$

Typically, an acquisition function offers an efficient trade-off between exploration of unsampled regions (high posterior variance) and exploitation of promising ones (low posterior mean), and has an analytical expression which makes it inexpensive to

evaluate, conveniently allowing to use of-the-shelf optimization algorithms to solve Eq. (10). In unconstrained, noise-free optimization, the canonical choice for $J(\mathbf{x})$ is the so-called *Expected Improvement* [EI, 36], while in the bandit literature (noisy observations), the Upper Confidence Bound [UCB, 58] can be considered as standard. Extensions abound in the literature to tackle various optimization problems: see e.g. Wagner et al [61] for multi-objective optimization or Hernández-Lobato et al [31] for constrained problems.

Figure 1 provides an illustration of the BO principles.

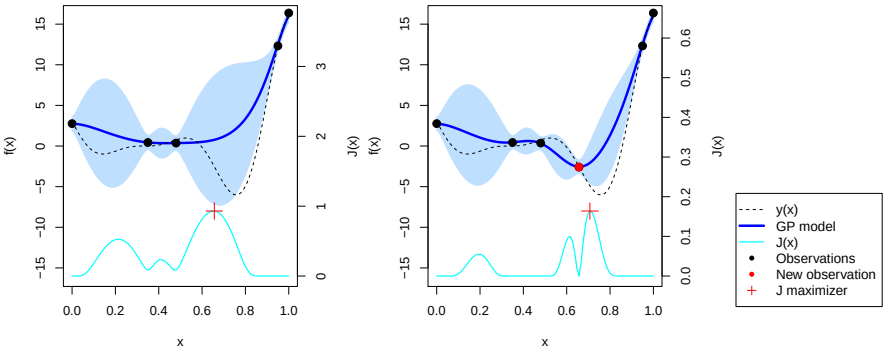

Fig. 1 One iteration of Bayesian optimization on a one-dimensional toy problem. A first GP is conditioned on a set of five observations (left), out of which an acquisition function is maximized to find the next observation. Once this observation is performed (right), the GP model and acquisition function are updated, and start pointing towards the optimum $x = 0.75$. Note that the acquisition is also large in unexplored regions (around $x = 0.2$).

3 Acquisition functions for NEP

We propose in the following two acquisition functions tailored to solve NEPs, respectively based on the probability of achieving equilibrium and on stepwise uncertainty reduction. Both aim at providing an efficient trade-off between exploration and exploitation.

3.1 Probability of equilibrium

Given a predictive distribution of $\mathbf{Y}(.)$, a first natural metric to consider is the probability of achieving the NE. From (2), using the notation $\mathbf{x} = (\mathbf{x}_i, \mathbf{x}_{-i})$, this probability writes:

$$\mathbb{P}_E(\mathbf{x}) = \mathbb{P}\left(\bigcap_{i=1}^{p}\left\{Y_i(\mathbf{x}_i, \mathbf{x}_{-i}) = \min_{\mathbf{x}_i^k \in \mathbb{X}_i} Y_i(\mathbf{x}_i^k, \mathbf{x}_{-i})\right\}\right), \tag{11}$$

where $\{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{m_i}\}$ denotes the m_i alternatives in $\mathbb{X}_i$, and $\mathbf{x}_{-i}$ is fixed to its value in $\mathbf{x}$.

Since our GP model assumes the independence of the posterior distributions of the objectives, we have:

$$\mathbb{P}_E(\mathbf{x}) = \prod_{i=1}^{p} \mathbb{P}\left\{ Y_i(\mathbf{x}) = \min_{\mathbf{x}_i^k \in \mathbb{X}_i} Y_i(\mathbf{x}_i^k, \mathbf{x}_{-i}) \right\} := \prod_{i=1}^{p} P_i(\mathbf{x}). \tag{12}$$

Let us now introduce the notation $\mathbf{x}_i = \mathbf{x}_i^l$ ($1 \leq l \leq m_i$). As exploited recently by Chevalier and Ginsbourger [8] in a multi-point optimization context, each P_i can be expressed as

$$P_i(\mathbf{x}) = \mathbb{P}\left(\bigcap_{k \in m_i, k \neq l} \left\{ Y_i(\mathbf{x}_i^l, \mathbf{x}_{-i}) - Y_i(\mathbf{x}_i^k, \mathbf{x}_{-i}) \leq 0 \right\} \right). \tag{13}$$

$P_i(\mathbf{x})$ amounts to compute the cumulative distribution function (CDF) of a Gaussian vector of size $q := m_i - 1$:

$$P_i(\mathbf{x}) = \mathbb{P}\left(\mathbf{Z}_i \leq \mathbf{0}\right) = \boldsymbol{\Phi}_{\boldsymbol{\mu}_{Z_i}, \boldsymbol{\Sigma}_{Z_i}}(\mathbf{0}), \tag{14}$$

with

$$\begin{aligned} \mathbf{Z}_i = \big[& Y_i(\mathbf{x}_i^1, \mathbf{x}_{-i}) - Y_i(\mathbf{x}_i^l, \mathbf{x}_{-i}), \ldots, Y_i(\mathbf{x}_i^{l-1}, \mathbf{x}_{-i}) - Y_i(\mathbf{x}_i^l, \mathbf{x}_{-i}), \\ & Y_i(\mathbf{x}_i^{l+1}, \mathbf{x}_{-i}) - Y_i(\mathbf{x}_i^l, \mathbf{x}_{-i}), \ldots, Y_i(\mathbf{x}_i^{m_i}, \mathbf{x}_{-i}) - Y_i(\mathbf{x}_i^l, \mathbf{x}_{-i}) \big]. \end{aligned}$$

The mean $\boldsymbol{\mu}_{Z_i}$ and covariance $\boldsymbol{\Sigma}_{Z_i}$ of $\mathbf{Z}_i$ can be expressed as:

$$\begin{aligned} (\mu_{Z_i})_j &= \mu_i(\mathbf{x}_i^l, \mathbf{x}_{-i}) - \mu_i(\mathbf{x}_i^j, \mathbf{x}_{-i}), \\ (\Sigma_{Z_i})_{jk} &= c_i^{ll} + c_i^{jk} - c_i^{lj} - c_i^{lk} \quad \text{if } k, l \neq j \text{ and } c_i^{ll} \text{ otherwise,} \end{aligned}$$

with $c_i^{jk} = \sigma_i^2 \left((\mathbf{x}_i^j, \mathbf{x}_{-i}), (\mathbf{x}_i^k, \mathbf{x}_{-i}) \right)$.

Several fast implementations of the multivariate Gaussian CDF are available, for instance in the `R` packages `mnormt` (for $q < 20$) [3] or, up to $q = 1000$, by Quasi-Monte-Carlo with `mvtnorm` [17, 18].

Alternatively, this quantity can be computed using Monte-Carlo methods by drawing R samples $\mathcal{Y}_i^{(1)}, \ldots, \mathcal{Y}_i^{(R)}$ of $\left[Y_i(\mathbf{x}_i^1, \mathbf{x}_{-i}), \ldots, Y_i(\mathbf{x}_i^{m_i}, \mathbf{x}_{-i}) \right]$, to compute

$$\hat{P}_i(\mathbf{x}) = \frac{1}{R} \sum_{r=1}^{R} \mathbb{1}\left(\mathcal{Y}_i^{(r)}(\mathbf{x}) = \min_{\mathbf{x}_i^k \in \mathbb{X}_i} \mathcal{Y}_i^{(r)}(\mathbf{x}_i^k, \mathbf{x}_{-i}) \right),$$

$\mathbb{1}(.)$ denoting the indicator function. This latter approach may be preferred when the number of alternatives m_i is high (say > 20), which makes the CDF evaluation overly expensive while a coarse estimation may be sufficient. Note that in both cases, a substantial computational speed-up can be achieved by removing from the $\mathbb{X}_i$'s the non-critical strategies. This point is discussed in Section 3.2.3.

Using $J(\mathbf{x}) = \mathbb{P}_E(\mathbf{x})$ as an acquisition function defines our first sequential sampling strategy. The strategy is rather intuitive: i.e., sampling at designs most likely to achieve NE.

Still, maximizing $\mathbb{P}_E$ is a *myopic* approach (i.e., favoring an immediate reward instead of a long-term one), which are often sub-optimal [see e.g. 20, 21, and references therein]. Instead, other authors have advocated the use of an information gain from a new observation instead, see e.g., Villemonteix et al [60], Hennig and Schuler [29], which motivated the definition of an alternative acquisition function, which we describe next.

3.2 Stepwise uncertainty reduction

Stepwise Uncertainty Reduction (SUR, also referred to as *information-based approach*) has recently emerged as an efficient approach to perform sequential sampling, with successful applications in optimization [60, 47, 30, 31] or uncertainty quantification [5, 34]. Its principle is to perform a sequence of observations in order to reduce as quickly as possible an uncertainty measure related to the quantity of interest (in the present case: the equilibrium).

3.2.1 Acquisition function definition

Let us first denote by $\Psi(\mathbf{y})$ the application that associates a NE with a multivariate function. In the case of finite games, we have: $\Psi : \mathbb{R}^{N\times p} \to \mathbb{R}^p$, for which a pseudo-code is detailed in Algorithm 3. If we consider the random process $\mathbf{Y}$ (Eq. 8) in lieu of the deterministic objective $\mathbf{y}$, the equilibrium $\Psi(\mathbf{Y})$ is a random vector of $\mathbb{R}^p$ with unknown distribution. Let Γ be a measure of variability (or residual uncertainty) of $\Psi(\mathbf{Y})$; we use here the determinant of its second moment:

$$\Gamma(\mathbf{Y}) = \det\left[\mathrm{cov}\left(\Psi(\mathbf{Y})\right)\right]. \tag{15}$$

The SUR strategy aims at reducing Γ by adding sequentially observations $\mathbf{y}(\mathbf{x})$ on which $\mathbf{Y}$ is conditioned. An "ideal" choice of $\mathbf{x}$ would be:

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}\in\mathbb{X}} \Gamma\left[\mathbf{Y}|\mathbf{f} = \mathbf{y}(\mathbf{x})\right], \tag{16}$$

where $\mathbf{Y}|\mathbf{f} = \mathbf{y}(\mathbf{x})$ is the process conditioned on the observation $\mathbf{f}$. Since we do not want to evaluate $\mathbf{y}$ for all $\mathbf{x}$ candidates, we consider the following criterion instead:

$$J(\mathbf{x}) = \mathbb{E}_{\mathbf{F}}\left(\Gamma\left[\mathbf{Y}|\mathbf{F} = \mathbf{Y}(\mathbf{x}) + \boldsymbol{\varepsilon}\right]\right), \tag{17}$$

with $\mathbf{F}$ following the posterior distribution (conditioned on the n current observations, Eq. 9) and $\mathbb{E}_{\mathbf{F}}$ denoting the expectation over $\mathbf{F}$.

In practice, computing $J(\mathbf{x})$ is a complex task, as no closed-form expression is available. The next subsection is dedicated to this question.

Remark For simplicity of exposition, we assume here that the equilibrium $\Psi(\mathbf{Y})$ exists, which is not guaranteed even if $\Psi(\mathbf{y})$ does. To avoid this problem, one may consider an extended $\bar{\Psi}$ function equal to $+\infty \times \mathbb{I}_p$ if there is no equilibrium and to Ψ otherwise, and in Eq. 15 use the restriction of $\bar{\Psi}$ to finite values.

3.2.2 Approximation using conditional simulations

Let us first focus on the measure Γ when no new observation is involved. Due to the strong non-linearity of Ψ, no analytical simplification is available, so we rely on conditional simulations of $\mathbf{Y}$ to evaluate Γ.

Let $\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M$ be independent draws of $\mathbf{Y}(\mathbb{X})$ (each $\boldsymbol{\mathcal{Y}}_i \in \mathbb{R}^{N \times p}$). For each draw, the corresponding NE $\Psi(\boldsymbol{\mathcal{Y}}_i)$ can be computed by exhaustive search. We reported to Appendix C the particular algorithm we used for this step. The following empirical estimator of $\Gamma(\mathbf{Y})$ is then available:

$$\hat{\Gamma}\left(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M\right) = \det\left[\mathbf{Q}_{\mathcal{Y}}\right],$$

with $\mathbf{Q}_{\mathcal{Y}}$ the sample covariance of $\Psi(\boldsymbol{\mathcal{Y}}_1), \ldots, \Psi(\boldsymbol{\mathcal{Y}}_M)$.

Now, let us assume that we evaluate the criterion for a given candidate observation point $\mathbf{x}$. Let $\mathcal{F}^1, \ldots, \mathcal{F}^K$ be independent draws of $\mathbf{F}(\mathbf{x}) = \mathbf{Y}(\mathbf{x}) + \boldsymbol{\varepsilon}$. For each $\mathcal{F}^i$, we can condition $\mathbf{Y}$ on the event $(\mathbf{F}(\mathbf{x}) = \mathcal{F}^i)$ in order to generate $\boldsymbol{\mathcal{Y}}_1|\mathcal{F}^i, \ldots, \boldsymbol{\mathcal{Y}}_M|\mathcal{F}^i$ draws of $\mathbf{Y}|\mathcal{F}^i$, from which we can compute the empirical estimator $\hat{\Gamma}\left(\boldsymbol{\mathcal{Y}}_1|\mathcal{F}^i, \ldots, \boldsymbol{\mathcal{Y}}_M|\mathcal{F}^i\right)$. Then, an estimator of $J(\mathbf{x})$ is obtained using the empirical mean:

$$\hat{J}(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{K} \hat{\Gamma}\left(\boldsymbol{\mathcal{Y}}_1|\mathcal{F}^i, \ldots, \boldsymbol{\mathcal{Y}}_M|\mathcal{F}^i\right).$$

3.2.3 Numerical aspects

The proposed SUR strategy has a substantial numerical cost, as the criterion requires a double loop for its computation: one over the K values of $\mathcal{F}^i$ and another over the M sample paths. The two computational bottlenecks are the sample path generations and the searches of equilibria, and both are performed in total $K \times M$ times for a single estimation of J. Thankfully, several computational shortcuts allow us to evaluate the criterion efficiently.

First, we employed the FOXY algorithm (*fast update of conditional simulation ensemble*) as proposed by Chevalier et al [9], in order to obtain draws of $\mathbf{Y}|\mathcal{F}^i$ based on a set of draws $\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M$. In short, a unique set of draws is generated prior to the search of $\mathbf{x}_{n+1}$, which is updated quickly when necessary depending on the pair $(\mathbf{x}, \mathcal{F}^i)$. The expression used are given in Appendix A, and we refer to [9] for the detailed algebra and complexity analysis.

Second, we discard points in $\mathbb{X}$ that are unlikely to provide information regarding the equilibrium prior to the search of $\mathbf{x}_{n+1}$, as we detail below. By doing so, we reduce substantially the sample paths size and the dimension of each finite game, which drastically reduces the cost. We call $\mathbb{X}_{\text{sim}}$ the retained subset.

Finally, $\hat{J}$ is evaluated only on a small, promising subset of $\mathbb{X}_{\text{sim}}$. We call $\mathbb{X}_{\text{cand}}$ this set.

To select the subsets, we rely on a fast-to-evaluate score function C, which can be seen as a proxy to the more expensive acquisition function. The subset of $\mathbb{X}$ is then chosen by sampling randomly with probabilities proportional to the scores $C(\mathbb{X})$, while ensuring that the subset retains a factorial form. We propose three scores, of increasing complexity and cost, which can be interleaved:

- C_{target}: the simplest score is the posterior density at a target T_E in the objective space, for instance the NE of the posterior mean (hence, it requires one NE search). C_{target} reflects a proximity to an estimate of the NE. We use this scheme for the first iteration to select $\mathbb{X}_{\text{sim}} \subset \mathbb{X}$.
- C_{box}: once conditional simulations have been performed, the above scheme can be replaced by the probability for a given strategy to fall into the box defined by the extremal values of the simulated NE (i.e., $\Psi(\boldsymbol{\mathcal{Y}}_1), \ldots, \Psi(\boldsymbol{\mathcal{Y}}_M)$). We use this scheme to select $\mathbb{X}_{\text{sim}} \subset \mathbb{X}$ for all the other iterations.
- $C_{\mathbb{P}}$: since $\mathbb{P}_E$ is faster (in particular in its Monte Carlo setting with small R) than $\hat{J}(\mathbf{x})$, it can be used to select $\mathbb{X}_{\text{cand}} \subset \mathbb{X}_{\text{sim}}$.

The detailed expressions of C_{target} and C_{box} are given in Appendix B. Note that in our experiments, C_{target} and C_{box} are also used with the $\mathbb{P}_E$ acquisition function.

Last but not least, this framework enjoys savings from parallelization in several ways. In particular, the searches of NE for each sample $\boldsymbol{\mathcal{Y}}$ can be readily distributed.

An overview of the full SUR approach is given in pseudo-code in Algorithm 1. Note that by construction, SUR does not necessarily sample at the NE, even when it is well-identified. Hence, as a post-processing step, the returned NE estimator is the design that maximizes the probability of achieving equilibrium.

Algorithm 1 Pseudo-code for the SUR approach

Require: $n_0, n_{\max}, N_{\text{sim}}, N_{\text{cand}}$
Construct initial design of experiments $\mathbf{X}_{n_0}$
Evaluate $\mathbf{y}_{n_0} = \mathbf{F}(\mathbf{X}_{n_0})$
while $n \leq n_{\max}$ **do**
  Train the p GP models on the current design of experiments $\{\mathbf{X}_n, \mathbf{y}_n\}$
  if $n = n_0$ **then**
    estimate $T_E = \Psi(\boldsymbol{\mu}(\mathbb{X}))$, the NE on the posterior mean; select $\mathbb{X}_{\text{sim}} \subset \mathbb{X}$ using C_{target}
  else
    Select $\mathbb{X}_{\text{sim}} \subset \mathbb{X}$ using C_{box}
  end if
  Generate M draws $(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M)$ on $\mathbb{X}_{\text{sim}}$
  Compute $\Psi(\boldsymbol{\mathcal{Y}}_1), \ldots, \Psi(\boldsymbol{\mathcal{Y}}_M)$ (for C_{box})
  Select $\mathbb{X}_{\text{cand}} \subset \mathbb{X}_{\text{sim}}$ using $C_{\mathbb{P}}$
  Find $\mathbf{x}_{n+1} = \arg\min_{\mathbf{x} \in \mathbb{X}_{\text{cand}}} \hat{J}(\mathbf{x})$
  Evaluate $\mathbf{y}_{n+1} = \mathbf{F}(\mathbf{x}_{n+1})$ and add $\{\mathbf{x}_{n+1}, \mathbf{y}_{n+1}\}$ to the current design of experiments
end while
Ensure: $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{X}_{\text{cand}}} \mathbb{P}_E(\mathbf{x})$

3.2.4 Stopping criterion

We consider in Algorithm 1 a fixed budget, assuming that simulation cost is limited. However, other natural stopping criteria are available. For $\mathbb{P}_E$, one may stop if there is a strategy for which the probability is high, i.e., $\max \mathbb{P}_E(\mathbf{x}) \geq 1 - \epsilon$ (with ϵ close to zero). For SUR, $\hat{J}$ is an indicator of the remaining uncertainty, so the algorithm may stop if this uncertainty is below a threshold, i.e., $\min J(\mathbf{x}) \leq \epsilon$.

While there always exists a Nash equilibrium for mixed strategy, in the setup we entertain there may actually be no pure strategy, or several. For $\mathbb{P}_E$, this is completely transparent, with a probability zero for all strategies (no NE) or several strategies having probability one (multiple NEs). For SUR, with more than one equilibrium, no change is needed, even though SUR may benefit from using a clustering method of the simulated NEs and defining local variability instead of the global Γ, Eq. (15). The absence of NE on the GP draws ($\Psi(\boldsymbol{\mathcal{Y}}_i)$) can be used to detect the absence of NE for the problem at hand.

4 NUMERICAL EXPERIMENTS

These experiments have been performed in `R` [50] using the `GPGame` package [48], which relies on the `DiceKriging` package [53] for the Gaussian process regression part.

We used a fixed-point method as a competitive alternative to compute Nash equilibria, in order to assess the efficiency of our approach. It is a popular method among the audience who is familiar with gradient-descent optimization algorithms. The algorithm pseudo-code is given in Algorithm 2, and has been implemented in `Scilab` [55].

Algorithm 2 Pseudo-code for the fixed-point approach [59]

Require: P: number of players, $0 < \alpha < 1$: relaxation factor, $k_{\max}$: max iterations
1: Construct initial strategy $\mathbf{x}^{(0)}$
2: **while** $k \leq k_{\max}$ **do**
3: Compute in parallel : $\forall i,\ 1 \leq i \leq P, \quad \mathbf{z}_i^{(k+1)} = \arg\min_{\mathbf{x}_i \in \mathbb{X}_i} J_i(x_{-i}^{(k)}, x_i)$
4: Update : $\mathbf{x}^{(k+1)} = \alpha \mathbf{z}^{(k+1)} + (1-\alpha) x^{(k)}$
5:
6: **if** $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$ small enough **then** exit
7: **end if**
8: **end while**
Ensure: For all $i = 1...P$, $\mathbf{x}_i^* = \arg\min_{\mathbf{x}_i \in \mathbb{X}_i} J_i(x_{-i}^*, x_i)$

In the following, performance is assessed in terms of numbers of calls to the objective function, hence assuming that the cost of running the black-box model largely exceeds the cost of choosing the points. For moderately expensive problems, the choice of algorithm may depend on the budget, as, intuitively, the time used to search for a new point may not exceed the time to simulate it. We report in Appendix D the computational times required for our approach on the three following test problems.

4.1 A classical multi-objective problem

We first consider a classical optimization toy problem (P1) as given in [46], with two variables and two cost functions, defined as:

$$y_1 = (x_2 - 5.1(x_1/(2\pi))^2 + \frac{5}{\pi}x_1 - 6)^2 + 10((1 - \frac{1}{8\pi})\cos(x_1) + 1)$$
$$y_2 = -\sqrt{(10.5 - x_1)(x_1 + 5.5)(x_2 + 0.5)} - \frac{(x_2 - 5.1(x_1/(2\pi))^2 - 6)^2}{30}$$
$$- \frac{(1 - 1/(8\pi))\cos(x_1) + 1}{3}$$

with $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. Both functions are non-convex. We set $\mathbf{x}_1 = x_1$ and $\mathbf{x}_2 = x_2$. The actual NE is attained at $x_1 = -3.786$ and $x_2 = 15$.

Our strategies are parameterized as follow. $\mathbb{X}$ is first discretized over a 31×31 regular grid. With such size, there is no need to resort to subsets, so we use $\mathbb{X}_{\text{cand}} = \mathbb{X}_{\text{sim}} = \mathbb{X}$. We set the number of draws to $K = M = 20$, which was empirically found as a good trade-off between accuracy and speed. We use $n_0 = 6$ initial observations from a Latin hypercube design [LHD, 42], and observations are added sequentially using both acquisition functions. As a comparison, we ran a standard fixed-point algorithm [4] based on finite differences. This experiment is replicated five times with different initial sets of observations for the GP-based approaches and different initial points for the fixed-point algorithm. The results are reported in Table 1.

In addition, Figure 2 provides some illustration for a single run. In the initial state (top left), the simulated NEs form a cloud in the region of the actual NE. A first point is added, although far from the NE, that impacts the form of the cloud (top, middle). The infill criterion surface (Figure 2, top right) then targets the top left corner, which offers the best compromise between exploration and exploitation. After adding 7 points (bottom left) the cloud is smaller and centered on the actual NE. As the NE is quite well-identified, the infill criterion surface (Figure 2, bottom right) is now relatively flat and mostly indicates regions that have not been explored yet. After 14 additions (bottom, middle), all simulated NE but two concentrate on the actual NE. The observed values have been added around the actual NE, but not only, which indicates that some exploration steps (such as iteration 8) have been performed.

Strategy	Evaluations required	Success rate
$\mathbb{P}_E$	9–10	5/5
SUR	8–14	5/5
Fixed point	200–1000	3/5

Table 1 P1 convergence results.

Both GP approaches consistently find the NE very quickly: the worst run required 14 cost evaluations (that is, 8 infill points). In contrast, the classical fixed-point algorithm required hundreds of evaluations, which is only marginally better than an exhaustive search on the 961-points grid. Besides, two out of five runs converged to the

stationary point $(1, 1)$, which is not a NE. On this example, $\mathbb{P}_E$ performed slightly better than SUR.

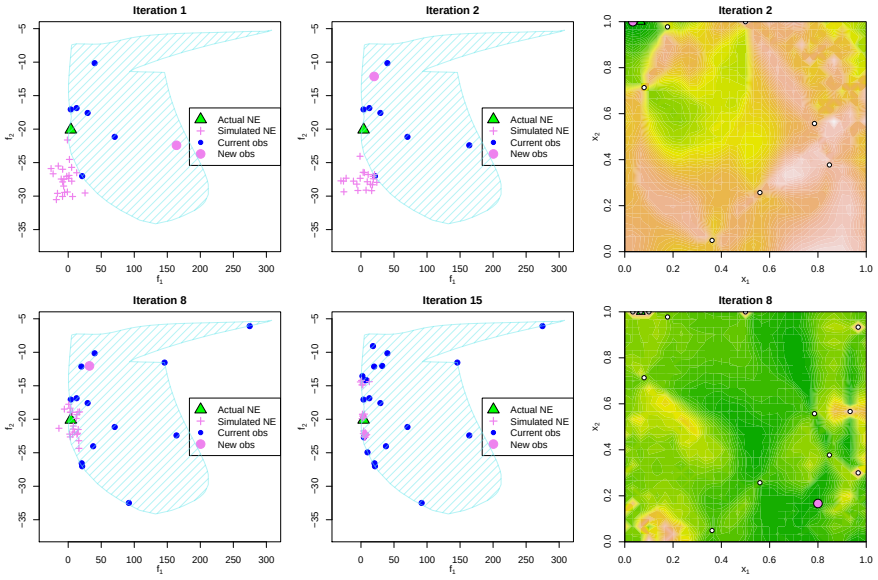

Fig. 2 Four iterations of SUR for (P1). Left and middle: observations and simulated NEs in the objective space. The hatched area represents the image of $\mathbb{X}$ by $\mathbf{y}$. Right: level sets of the SUR criterion value in the variable space (green is better).

4.2 An open loop differential game

Differential games model a huge variety of competitive interactions, in social behavior, economics, biology among many others [predator-prey, pursuit-evasion games and so on, see 33]. As a toy-model, let us consider p players who compete to drive a dynamic process to their preferred final state. Let denote $T > 0$ the final time, and $t \in [0, T]$ the time variable. Then we consider the following simple process, which state $\mathbf{z}(t) \in \mathbb{R}^2$ obeys the first order dynamics:

$$\begin{cases} \dot{\mathbf{z}}(t) = \mathbf{v}_0 + \sum_{1 \leq i \leq p} \alpha_i(t) \mathbf{x}_i(t) \\ \mathbf{z}(0) = \mathbf{z}_0 \end{cases} \tag{18}$$

The parameter $\alpha_i(t) = e^{-\theta_i t}$ models the lifetime range of each player's influence on the process (θ_i is a measure of player (i)'s preference for the present). The time-dependent action $\mathbf{x}_i(t) \in \mathbb{R}^2$ is the player (i)'s strategy, which we restrict to the (finite-dimensional) space of spline functions of a given degree $\kappa - 1$, that is:

$$\mathbf{x}_i(t) = \begin{bmatrix} \sum_{1 \leq k \leq \kappa} a_k^{(i)} \times B_k(t/T) \\ \sum_{1 \leq k \leq \kappa} b_k^{(i)} \times B_k(t/T) \end{bmatrix},$$

where $(B_k)_{1\leq k\leq \kappa}$ is the spline basis.

Now, the decision variables for player (i) is the array $(a_1^{(i)},\dots,a_\kappa^{(i)},b_1^{(i)},\dots,b_\kappa^{(i)})$ (the spline coefficients), and $\mathbf{x}\in\mathbb{R}^{\kappa\times 2\times p}$. We used $\kappa=1$ (constant splines, two decision variables per player) and $\kappa=2$ (linear splines, four decision variables per player).

All players have their own preferred final states $\mathbf{z}_T^i\in\mathbb{R}^2$, and a limited energy to dispense in playing with their own action. The cost y_i of player (i) is then the following :

$$y_i(\mathbf{x}_i,\mathbf{x}_{-i})=\frac{1}{2}\|\mathbf{z}(T)-\mathbf{z}_T^i\|_{\mathbb{R}^2}^2+\frac{1}{2}\|\mathbf{x}_i\|_{L^2(0,T)}^2. \tag{19}$$

The game considered here is an open loop differential game (the decisions do not depend on the state), and belongs to the larger class of dynamic Nash games.

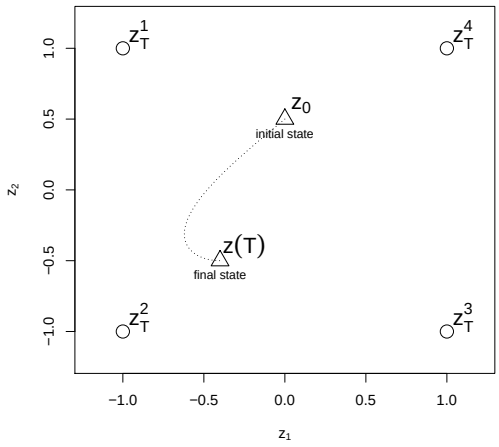

Fig. 3 The differential game setting: The process starts at the initial state $\mathbf{z}_0$ and travels during a time T to reach the state $\mathbf{z}(T)$. Players targets are $\mathbf{z}_T^{(i)}$.

When the α_i's do not depend on i, and the points $\mathbf{z}_T^i$ are located on some -any- circle, then the Nash solution of the game puts the final state $\mathbf{z}(T)$ on the center of the circle (provided T is large enough to let the state reach this center starting from z_0). In our setup, we have $p=4$ players, with targets positioned on the four corners of the $[-1,1]^2$ square, as illustrated by Figure 3. The state equation (18) is solved by means of an explicit Euler numerical scheme, with T set to 4, and 40 time steps. As for problem parameters, we chose $\mathbf{v}_0=\mathbf{0}$, $\mathbf{z}_0=(0,0.5)$ (non-central initial position) and $\boldsymbol{\theta}=(\theta_i)=(0.25,0,0.5,0)$ (heterogeneous time preferences), which makes the search of NE non-trivial. The results labeled "Fixed point" presented in Table 2 are obtained by means of a popular fixed-point algorithm, as described in Algorithm-2.

The initial design space is set as $[-6,6]^d$ (with $d=8$ or $d=16$) and discretized as follow. For each set of design variables belonging to a player, we first generate

Configuration		Strategy	Evaluations required
$d = 8$	$(\kappa = 1)$	$\mathbb{P}_E$	83–95
$d = 8$	$(\kappa = 1)$	SUR	81–88
$d = 8$	$(\kappa = 1)$	Fixed point	3000–5000
$d = 16$	$(\kappa = 2)$	$\mathbb{P}_E$	196–221
$d = 16$	$(\kappa = 2)$	SUR	208–232
$d = 16$	$(\kappa = 2)$	Fixed point	5000–7000

Table 2 Differential game convergence results.

a 17-point LHD. For $\kappa = 1$, this means four LHD in the spaces (x_1, x_2), (x_3, x_4) and so on. Then, we take all the combinations of strategies, ending with a total of $\text{Card}(\mathbb{X}) = N = 17^4 = 83,521$ possible strategies.

In both cases, we followed Algorithm 1 with $n_0 = 80$ or 160 initial design points and a maximum budget of $n_{\max} = 160$ or 320 evaluations (depending on the dimension). We chose $\text{Card}(\mathbb{X}_{\text{sim}}) = N_{\text{sim}} = 1296$ simulation points and $\text{Card}(\mathbb{X}_{\text{cand}}) = N_{\text{cand}} = 256$ candidates. For the number of draws to compute the SUR criterion, we chose $K = M = 20$.

A fixed-point algorithm based on finite differences is also ran, and the experiment is replicated five times with different algorithm initializations. Note that, here, the actual NE is unknown beforehand. However, since all runs (including the fixed-point algorithm ones) converged to the same point, we assume that it is the actual NE.

We show the results in Table 2. For the lower dimension problem ($d = 8$), all runs found the solution with less than 100 evaluations (that is, 80 initial points plus 20 infills). This represents about 1% of the total number of possible strategies. In this case, SUR appears as slightly more efficient than $\mathbb{P}_E$. For the higher dimension problem ($d = 16$), more evaluations are needed, yet all runs converge with less than 240 evaluations. As a comparison, on both cases the fixed-point algorithm is more than 60 (resp. 20) times more expensive.

4.3 PDE-constrained example: data completion

4.3.1 Problem description

We address here the class of problems known as data completion or data recovery problems. Let be Ω a bounded open domain in $\mathbb{R}^d$ ($d = 2, 3$) with a sufficiently smooth boundary $\partial\Omega$ composed of two connected disjoint components Γ_c and Γ_i, with the latter being inaccessible to boundary measurements. For details, see [24] whence the present example is excerpt.

Let us focus, for illustration, on the particular case of steady state heat equation. The problem is formulated in terms of the following elliptic Cauchy problem :

$$\begin{cases} \nabla.(\lambda \nabla u) = 0 \text{ in } \Omega \\ u = \varphi \text{ on } \Gamma_c \\ \lambda \nabla u.\nu = \Phi \text{ on } \Gamma_c \end{cases} \tag{20}$$

The data to be recovered, or missing data, are $u_{|\Gamma_i}$ and $\lambda\nabla u.\nu_{|\Gamma_i}$, which are determined as soon as one knows u in the whole Ω. The parameters λ, φ and Φ are given functions, ν is the unit outward normal vector on the boundary. The Dirichlet data φ and the Neumann data Φ are the so-called Cauchy data, which are known on the accessible part Γ_c of the boundary $\partial\Omega$ and the unknown field u is the Cauchy solution. Completion/Cauchy problems are known to be severely ill-posed (Hadamard's), and computationally challenging.

Let us assume that $(\Phi, \varphi) \in H^{-\frac{1}{2}}(\Gamma_c) \times H^{\frac{1}{2}}(\Gamma_c)$ where $H^{-\frac{1}{2}}(\Gamma_c)$ resp. $H^{\frac{1}{2}}(\Gamma_c)$ are the Sobolev spaces of Neumann resp. Dirichlet traces of functions in the Sobolev space $H^1(\Omega)$ [see e.g. 1]. For given $\eta \in H^{-\frac{1}{2}}(\Gamma_i)$ and $\zeta \in H^{\frac{1}{2}}(\Gamma_i)$, let us define $u_1(\eta)$ and $u_2(\zeta)$ as the unique solutions in $H^1(\Omega)$ of the following elliptic boundary value problems :

$$(SP1)\begin{cases} \nabla.(\lambda\nabla u_1) = 0 \text{ in } \Omega \\ u_1 = \varphi \text{ on } \Gamma_c \\ \lambda\nabla u_1.\nu = \eta \text{ on } \Gamma_i \end{cases} \quad (SP2)\begin{cases} \nabla.(\lambda\nabla u_2) = 0 \text{ in } \Omega \\ u_2 = \zeta \text{ on } \Gamma_i \\ \lambda\nabla u_2.\nu = \Phi \text{ on } \Gamma_c \end{cases} \tag{21}$$

Let us define the following two costs : for any $\eta \in H^{-\frac{1}{2}}(\Gamma_i)$ and $\zeta \in H^{\frac{1}{2}}(\Gamma_i)$,

$$J_1(\eta,\zeta) = \frac{1}{2}\|\lambda\nabla u_1.\nu - \Phi\|^2_{H^{-\frac{1}{2}}(\Gamma_c)} + \frac{\alpha}{2}\|u_1 - u_2\|^2_{H^{\frac{1}{2}}(\Gamma_i)}, \tag{22}$$

$$J_2(\eta,\zeta) = \frac{1}{2}\|u_2 - \varphi\|^2_{L^2(\Gamma_c)} + \frac{\alpha}{2}\|u_1 - u_2\|^2_{H^{\frac{1}{2}}(\Gamma_i)}, \tag{23}$$

where α is a given positive parameter (e.g. $\alpha = 1$).

Let us remark that, for the problem is severely ill posed, the classical descent algorithms for the minimization of the cost $J_1 + J_2$ with respect to the couple of variables (η, ζ) do not converge. Besides, the solution is extremely sensitive to small variations of ϕ and Φ. This originally motivated the above-described game formulation.

The fields $u_1 = u_1(\eta)$ and $u_2 = u_2(\zeta)$ are aiming at the fulfillment of a possibly antagonistic goals, namely minimizing the Neumann gap $\|\lambda\nabla u_1.\nu - \Phi\|_{H^{-\frac{1}{2}}(\Gamma_c)}$ and the Dirichlet gap $\|u_2 - \varphi\|_{L^2(\Gamma_c)}$. This antagonism is intimately related to Hadamard's ill-posedness character of the Cauchy problem, and rises as soon as one requires that u_1 and u_2 coincide, which is exactly what the coupling term $\|u_1 - u_2\|_{L^2(\Gamma_i)}$ is for. Thus, one may think of an iterative process which minimizes in a smart fashion the three terms, namely Neumann-Dirichlet-Coupling terms.

From a game theory perspective, one may define two players, one associated with the strategy variable η and cost $y_1 = J_1$ and the second with the variable ζ and cost $y_2 = J_2$, each trying to minimize its cost in a non-cooperative fashion. The fact that each player controls only his own strategy, while there is a strong dependence of each player's cost on the joint strategies (η, ζ) justifies the use of the game theory framework (and terminology), a natural setting which may be used to formulate the negotiation between these two costs.

Theorem 1 *[24] There always exists a unique Nash equilibrium* $(\eta^*, \zeta^*) \in H^{-\frac{1}{2}}(\Gamma_i) \times H^{\frac{1}{2}}(\Gamma_i)$*, and when the Cauchy problem has a solution* u*, then* $u_1(\eta^*) = u_2(\zeta^*) = u$*, and* (η^*, ζ^*) *are the missing data, namely* $\eta^* = \lambda\nabla u.\nu_{|\Gamma_i}$ *and* $\zeta^* = u_{|\Gamma_i}$.

4.3.2 Noisy data and random Nash equilibrium

In a realistic situation, the fields φ and Φ may be "polluted" by noise. Habbal and Kallel [24] showed that the Nash equilibrium is stable with respect to small perturbations, and that the perturbed equilibrium converges strongly to the unperturbed one when the noise tends to zero. However, they did not consider directly the random Nash game:

$$\begin{cases} \min_\eta \mathbb{E}[J_1(\eta, \zeta, \varphi^\epsilon, \Phi^\epsilon)] \\ \min_\zeta \mathbb{E}[J_2(\eta, \zeta, \varphi^\epsilon, \Phi^\epsilon)], \end{cases} \tag{24}$$

where φ^ϵ and Φ^ϵ are perturbed values of φ and Φ.

Addressing this problem with a classical fixed-point algorithm *à la* [4] would require, for each trial pair (η, ζ), repeated evaluations of J_1 and J_2 for many different values of φ^ϵ and Φ^ϵ, which would prove extremely intensive computationally.

4.3.3 Implementation and experimental setup

The physical domain Ω is taken as a 2D annular structure. The accessible boundary Γ_c is the outer circle, with a ray $R_c = 1$ and the inaccessible boundary Γ_i is the inner circle with a ray $R_i = 0.5$, see Figure 4. The conductivity coefficient is $\lambda = 1$, the flux $\Phi = 0$ and the heat field φ is built from an exact known solution (an academic $u(x, y) = \exp(x)\cos(y)$ used for validation issues).

We use FreemFem++ [28] to develop our finite element (FE) solvers. The FE computations are performed with a $P1$-triangular mesh yielding $1,088$ degrees of freedom, the outer and inner boundaries being discretized each with 60 finite element nodes. From now on, we use the same notations as above for functions to refer to their finite element approximations (values at FE nodes).

As in Section 4.2, to reduce the dimensionality of the problem, the η and ζ being originally vectors of size 60 (the number of nodes at the inner boundary), we interpolate the underlying functions with -natural- splines with $\kappa = 8$ coefficients for each quantity, resulting with a decision space of size $d = 16$ (instead of the original 120). Each coefficient is bounded between -3 and 3, allowing a large variety of spline shapes.

Since the Neumann condition involving Φ is known to be the most sensitive to noise, we perturb only this term with an additive noise:

$$\Phi^\epsilon = \Phi_0 + \epsilon(\xi),$$

with $\epsilon(\xi)$ a white noise uniformly distributed between -0.25 and 0.25, thus resulting in a randomized vector of dimension 60 (the number of FE nodes on the outer boundary). The resulting variability on $J_1(\eta, \zeta, \Phi^\epsilon)$ and $J_2(\eta, \zeta, \Phi^\epsilon)$ strongly depends on the value of (η, ζ) and can be very large (with coefficients of variation, y_i/τ_i, from 5% to over 100%).

The original continuous space of the spline coefficients is discretized by generating first two 1000×8 LHD and taking all the combinations between the two designs, ending with space of size $N = 10^6$ potential strategies. The Bayesian optimization

strategy is run with $n_0 = 50$ initial points (chosen by space-filling design) and 150 infill points. Since the noise is very heteroskedastic, we use five repetitions for each trial pair (η, ζ) in order to obtain a rough estimate of the noise variance τ_i, and we take the mean over those five repetitions as our observations (f_i's). This results with a total budget of $(50+150)\times 5 = 1,000$ model runs. Let us notice that computing a *deterministic* NE with fixed-point methods requires about five times this budget on this problem. We followed Algorithm 1 with the SUR criterion, $K = M = 20$, selecting $\mathrm{Card}(\mathbb{X}_{\mathrm{sim}}) = N_{\mathrm{sim}} = 1,296$ simulation points and $\mathrm{Card}(\mathbb{X}_{\mathrm{cand}}) = N_{\mathrm{cand}} = 256$ candidates.

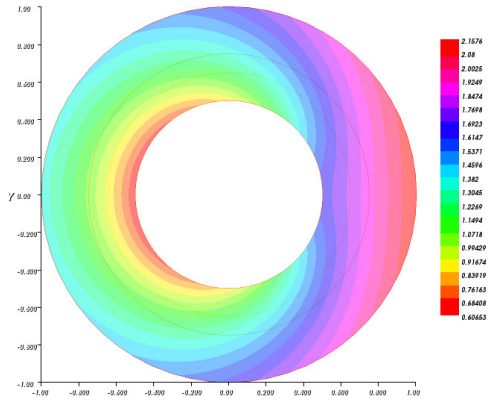

Fig. 4 The domain is an annular structure, the outer circle is the accessible boundary while the inner circle is the inaccessible one. The plot shows the isovalues of the solution u_1 at convergence.

4.3.4 Results

The algorithm is run five times with different initial points to assess its robustness. The results on the test problem are presented in Figure 5. In the absence of reference (*ground truth* to be compared to), we show the evolution of the two objectives functions during optimization. All five runs identify very quickly (after 20 iterations, which corresponds to 350 FE evaluations) a similar estimation of the equilibrium, close to the equilibrium of the noiseless problem (which is actually known to be $(0, 0)$, Habbal and Kallel [24]). However, it fails at converging finely to a single, stable value, in particular for y_2. This is not surprising and can largely be attributed to the difficulty of the problem (with respect to the noise level and dimensionality). Hybrid approaches (using fixed-point strategies for a final local convergence) or sophisticated sampling strategies (increasing the number of repeated simulations gradually with the iterations) may be used to solve this issue (although at the price of a considerably higher computational budget).

5 CONCLUDING COMMENTS

We have proposed here a novel approach to solve random or deterministic Nash games with drastically limited budgets of evaluations based on GP regression, taking the form of a Bayesian optimization algorithm. Experiments on challenging synthetic problems demonstrate the potential of this approach compared to classical, derivative-based algorithms.

On the test problems, the two acquisition functions performed similarly well. $\mathbb{P}_E$ has the benefit of not relying on conditional simulation paths, which makes it simpler to implement and less computationally intensive in most cases. Still, the SUR approach has several decisive advantages; in particular, it does not actually require the new observations to belong to the grid $\mathbb{X}_{\text{sim}}$, such that it could be optimized continuously. Moreover, it lays the groundwork for many extensions that may be pursued in future work.

First, SUR strategies are well-suited to allow selecting batches of points instead of only one, a key feature in distributed computer experiments [8]. Second, other games and equilibria may be considered: the versatility of the SUR approach may allow its transposition to other frameworks. In particular, mixed equilibria, which existence are always guaranteed on discrete problems, could be addressed by using Ψ functions that return discrete probability measures. Generalized NEPs [12] could be tackled by building on existing works on Bayesian optimization with constraints [see e.g., 31, and references therein].

Finally, this work could also be extended to continuous domains, for which related convergence properties may be investigated in light of recent advances on SUR approaches [6].

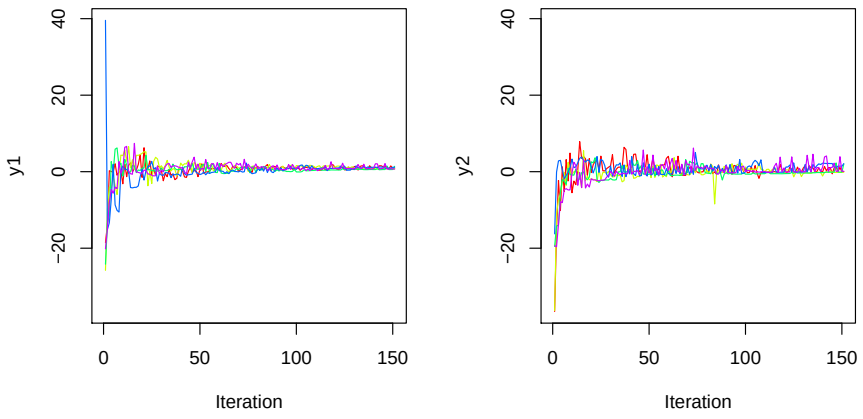

Fig. 5 Convergence of the estimated Nash equilibrium for five algorithm runs. Each iteration corresponds to five calls of the FreeFem++ model.

A Handling conditional simulations

We detail here how we generate the draws of $\mathbf{Y}|\boldsymbol{\mathcal{F}}_i$ to compute $\hat{J}(\mathbf{x})$ in practice. We employ the FOXY (*fast update of conditional simulation ensemble*) algorithm proposed by Chevalier et al [9], as detailed below.

Let $\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M$ be independent draws of $\mathbf{Y}$ ($\mathbb{X}$) (each $\boldsymbol{\mathcal{Y}}_i \in \mathbb{R}^{N\times p}$), generated using the posterior Gaussian distribution of Eq. (8), and $\boldsymbol{\mathcal{F}}_1, \ldots, \boldsymbol{\mathcal{F}}_K$ independent (of each other and of the $\boldsymbol{\mathcal{Y}}_i$'s) draws of $\mathbf{Y}(\mathbf{x}) + \boldsymbol{\varepsilon}$ from the posterior Gaussian distribution of Eq. (9). As shown in Chevalier et al [9], draws of $\mathbf{Y}|\boldsymbol{\mathcal{F}}_i$ can be obtained efficiently from $\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M$ using:

$$\mathcal{Y}_j^{(i)}|\mathcal{F}_k^{(i)} = \mathcal{Y}_j^{(i)} + \boldsymbol{\lambda}^{(i)}(\mathbf{x})\left(\mathcal{F}_k^{(i)} - \mathcal{Y}_j^{(i)}(\mathbf{x})\right), \tag{25}$$

with $1 \leq i \leq p$, $1 \leq j \leq M$, $1 \leq k \leq K$ and

$$\boldsymbol{\lambda}^{(i)}(\mathbf{x}) = \frac{\mathbf{k}_n^{(i)}(\mathbf{x}, \mathbb{X})}{\mathbf{k}_n^{(i)}(\mathbf{x}, \mathbf{x})}.$$

Notice that $\boldsymbol{\lambda}^{(i)}(\mathbf{x})$ may only be computed once for all $\mathcal{Y}_j^{(i)}(\mathbf{x})$.

B $C(\mathbf{x})$ formulae

For a given target $T_E \in \mathbb{R}^p$ and $\mathbf{x} \in \mathbb{X}$:

$$C_{\text{target}}(\mathbf{x}) = \prod_{i=1}^{p} \phi\left(\frac{T_{Ei} - \mu_i(\mathbf{x})}{\sigma_i(\mathbf{x})}\right), \tag{26}$$

with ϕ the probability density function of the standard Gaussian variable.

Let $T_L \in \mathbb{R}^p$ and $T_U \in \mathbb{R}^p$ such that $\forall 1 \leq i \leq p, T_{Li} < T_{Ui}$ define a box in the objective space. Defining $\boldsymbol{\Psi} = [\Psi(\boldsymbol{\mathcal{Y}}_1), \ldots, \Psi(\boldsymbol{\mathcal{Y}}_M)]$ the $p \times M$ matrix of simulated NE, we use:

$$\forall 1 \leq i \leq p \qquad T_{Li} = \min \boldsymbol{\Psi}_{i,1\ldots M} \quad \text{and} \quad T_{Ui} = \max \boldsymbol{\Psi}_{i,1\ldots M}.$$

Then, the probability to belong to the box is:

$$C_{\text{box}}(\mathbf{x}) = \prod_{i=1}^{p}\left[\Phi\left(\frac{T_{Ui} - \mu_i(\mathbf{x})}{\sigma_i(\mathbf{x})}\right) - \Phi\left(\frac{\mu_i(\mathbf{x}) - T_{Li}}{\sigma_i(\mathbf{x})}\right)\right]. \tag{27}$$

C Solving NEP on GP draws

We detail here a simple algorithm to extract Nash equilibria from GP draws.

D Computational time

We report here the computational time required to perform a single iteration of our algorithm for each of the three examples (not including the time required to run the simulation itself). Experiments were run on an Intel®Core™ i7-5600U CPU at 2.60GHz with 4 × 8GB of RAM.

Acknowledgements The authors acknowledge inspiration from Lorentz Center Workshop "SAMCO - Surrogate Model Assisted Multicriteria Optimization", at Leiden University Feb 29 - March 4, 2016. Mickaël Binois is grateful for support from National Science Foundation grant DMS-1521702.

Algorithm 3 Pseudo-code for Nash equilibria extraction

Require: P: number of players, $\mathcal{Y}$: draw of $\mathbf{Y}(\mathbb{X})$ of size $N \times P$, $I = \{1, \ldots, N\}$
for $1 \leq i \leq N$ **do**
  if $i \in I$ **then**
    for $1 \leq j \leq P$ **do**
      Find $K = \left\{1 \leq k \leq N, \mathbf{x}^k_{-j} = \mathbf{x}^i_{-j}\right\}$
      Find $l^* = \min_{1 \leq l \leq |K|} \mathcal{Y}(\mathbf{x}^l)_j$
      for $1 \leq l \leq |K|$ **do**
        if $\mathcal{Y}(\mathbf{x}^l)_j > \mathcal{Y}(\mathbf{x}^{l^*})_j$ **then**
          $I \leftarrow I \setminus l$
        end if
      end for
    end for
  end if
end for
Ensure: I

Case / Criterion	Pnash	SUR
Case 4.1	4s	20s
Case 4.2 ($\kappa = 1$)	13s	40s
Case 4.2 ($\kappa = 2$)	28s	82s
Case 4.3	9s	12s

Table 3 Average CPU times required for one iteration of the GP-based algorithm on the different test problems.

References

1. Adams RA, Fournier JJ (2003) Sobolev spaces, vol 140. Academic press
2. Álvarez MA, Rosasco L, Lawrence ND (2011) Kernels for vector-valued functions: A review. Foundations and Trends in Machine Learning 4(3):195–266, DOI 10.1561/2200000036, URL `http://dx.doi.org/10.1561/2200000036`
3. Azzalini A, Genz A (2016) The R package `mnormt`: The multivariate normal and t distributions (version 1.5-4). URL `http://azzalini.stat.unipd.it/SW/Pkg-mnormt`
4. Başar T (1987) Relaxation techniques and asynchronous algorithms for on-line computation of non-cooperative equilibria. J Econom Dynam Control 11(4):531–549
5. Bect J, Ginsbourger D, Li L, Picheny V, Vazquez E (2012) Sequential design of computer experiments for the estimation of a probability of failure. Statistics and Computing 22(3):773–793
6. Bect J, Bachoc F, Ginsbourger D (2016) A supermartingale approach to gaussian process based sequential design of experiments. arXiv preprint arXiv:160801118
7. Brown N, Ganzfried S, Sandholm T (2015) Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas hold'em agent. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp 7–15
8. Chevalier C, Ginsbourger D (2013) Fast computation of the multi-points expected improvement with applications in batch selection. In: Learning and Intelligent Optimization, Springer, pp 59–69
9. Chevalier C, Emery X, Ginsbourger D (2015) Fast update of conditional simulation ensembles. Mathematical Geosciences 47(7):771–789
10. Cressie N (1993) Statistics for spatial data: Wiley series in probability and statistics
11. Dorsch D, Jongen HT, Shikhman V (2013) On structure and computation of generalized nash equilibria. SIAM Journal on Optimization 23(1):452–474
12. Facchinei F, Kanzow C (2010) Generalized nash equilibrium problems. Annals of Operations Research 175(1):177–211

13. Fleuret F, Geman D (1999) Graded learning for object detection. In: Proceedings of the workshop on Statistical and Computational Theories of Vision of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR/SCTV), vol 2
14. Friedman A (1972) Stochastic differential games. Journal of differential equations 11(1):79–108
15. Games ILSC (2016) Lenient learning in independent-learner stochastic cooperative games. Journal of Machine Learning Research 17:1–42
16. Garivier A, Kaufmann E, Koolen WM (2016) Maximin action identification: A new bandit framework for games. In: 29th Annual Conference on Learning Theory, pp 1028–1050
17. Genz A, Bretz F (2009) Computation of Multivariate Normal and t Probabilities. Lecture Notes in Statistics, Springer-Verlag, Heidelberg
18. Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2016) mvtnorm: Multivariate Normal and t Distributions. URL `http://CRAN.R-project.org/package=mvtnorm`, r package version 1.0-5
19. Gibbons R (1992) Game Theory for Applied Economists. Princeton University Press, Princeton, NJ
20. Ginsbourger D, Le Riche R (2010) Towards Gaussian process-based optimization with finite time horizon. In: mODa 9–Advances in Model-Oriented Design and Analysis, Springer, pp 89–96
21. Gonzalez J, Osborne M, Lawrence N (2016) Glasses: Relieving the myopia of Bayesian optimisation. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, pp 790–799
22. Gramacy RB, Apley DW (2015) Local gaussian process approximation for large computer experiments. Journal of Computational and Graphical Statistics 24(2):561–578
23. Gramacy RB, Ludkovski M (2015) Sequential design for optimal stopping problems. SIAM Journal on Financial Mathematics 6(1):748–775
24. Habbal A, Kallel M (2013) Neumann-Dirichlet Nash strategies for the solution of elliptic Cauchy problems. SIAM J Control Optim 51(5):4066–4083, DOI 10.1137/120869808, URL `http://dx.doi.org/10.1137/120869808`
25. Habbal A, Petersson J, Thellner M (2004) Multidisciplinary topology optimization solved as a Nash game. Int J Numer Meth Engng 61:949–963
26. Harsanyi JC (1973) Games with randomly disturbed payoffs: A new rationale for mixed-strategy equilibrium points. International Journal of Game Theory 2(1):1–23
27. Heaton MJ, Datta A, Finley A, Furrer R, Guhaniyogi R, Gerber F, Gramacy RB, Hammerling D, Katzfuss M, Lindgren F, et al (2017) Methods for analyzing large spatial data: A review and comparison. arXiv preprint arXiv:171005013
28. Hecht F, Pironneau O, Le Hyaric A, Ohtsuka K (2010) Freefem++ v. 2.11. User?s Manual University of Paris 6
29. Hennig P, Schuler CJ (2012) Entropy search for information-efficient global optimization. The Journal of Machine Learning Research 13:1809–1837
30. Hernández-Lobato JM, Hoffman MW, Ghahramani Z (2014) Predictive entropy search for efficient global optimization of black-box functions. In: Advances in neural information processing systems, pp 918–926
31. Hernández-Lobato JM, Gelbart MA, Adams RP, Hoffman MW, Ghahramani Z (2016) A general framework for constrained bayesian optimization using information-based search. Journal of Machine Learning Research 17(160):1–53
32. Hu J, Wellman MP (2003) Nash q-learning for general-sum stochastic games. Journal of Machine learning research 4(Nov):1039–1069
33. Isaacs R (1965) Differential games. A mathematical theory with applications to warfare and pursuit, control and optimization. John Wiley & Sons, Inc., New York-London-Sydney
34. Jala M, Lévy-Leduc C, Moulines É, Conil E, Wiart J (2016) Sequential design of computer experiments for the assessment of fetus exposure to electromagnetic fields. Technometrics 58(1):30–42
35. Johanson M, Bowling MH (2009) Data biased robust counter strategies. In: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS), pp 264–271
36. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. Journal of Global optimization 13(4):455–492
37. Kanzow C, Steck D (2016) Augmented lagrangian methods for the solution of generalized nash equilibrium problems. SIAM Journal on Optimization 26(4):2034–2058
38. Lanctot M, Burch N, Zinkevich M, Bowling M, Gibson RG (2012) No-regret learning in extensive-form games with imperfect recall. In: Proceedings of the 29th International Conference on Machine Learning (ICML-12), pp 65–72

39. León ER, Pape AL, Désidéri JA, Alfano D, Costes M (2014) Concurrent aerodynamic optimization of rotor blades using a nash game method. Journal of the American Helicopter Society
40. Li S, Başar T (1987) Distributed algorithms for the computation of noncooperative equilibria. Automatica J IFAC 23(4):523–533
41. Littman ML, Stone P (2005) A polynomial-time nash equilibrium algorithm for repeated games. Decision Support Systems 39(1):55–66
42. McKay MD, Beckman RJ, Conover WJ (1979) Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2):239–245
43. Mockus J (1989) Bayesian Approach to Global Optimization: Theory and Applications. Springer
44. Neyman A, Sorin S (2003) Stochastic games and applications, vol 570. Springer Science & Business Media
45. Nishimura R, Hayashi S, Fukushima M (2009) Robust nash equilibria in n-person non-cooperative games: Uniqueness and reformulation. Pacific Journal of Optimization 5(2):237–259
46. Parr JM (2012) Improvement criteria for constraint handling and multiobjective optimization. PhD thesis, University of Southampton
47. Picheny V (2014) A stepwise uncertainty reduction approach to constrained global optimization. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, JMLR W&CP, vol 33, pp 787–795
48. Picheny V, Binois M (2017) GPGame: Solving Complex Game problems using Gaussian processes. URL `http://CRAN.R-project.org/package=GPGame`, r package version 0.1.3
49. Plumlee M (2014) Fast prediction of deterministic functions using sparse grid experimental designs. Journal of the American Statistical Association 109(508):1581–1591
50. R Core Team (2016) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL `https://www.R-project.org/`
51. Rasmussen CE, Williams C (2006) Gaussian Processes for Machine Learning. MIT Press, URL `http://www.gaussianprocess.org/gpml/`
52. Rosenmüller J (1971) On a generalization of the lemke–howson algorithm to noncooperative n-person games. SIAM Journal on Applied Mathematics 21(1):73–79
53. Roustant O, Ginsbourger D, Deville Y (2012) DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. Journal of Statistical Software 51(1):1–55, URL `http://www.jstatsoft.org/v51/i01/`
54. Rullière D, Durrande N, Bachoc F, Chevalier C (2016) Nested kriging predictions for datasets with a large number of observations. Statistics and Computing pp 1–19
55. Scilab Enterprises (2012) Scilab: Free and Open Source software for numerical computation. Scilab Enterprises, Orsay, France, URL `http://www.scilab.org`
56. Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N (2016) Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE 104(1):148–175
57. Shapley LS (1953) Stochastic games. Proceedings of the national academy of sciences 39(10):1095–1100
58. Srinivas N, Krause A, Kakade SM, Seeger M (2012) Information-theoretic regret bounds for gaussian process optimization in the bandit setting. Information Theory, IEEE Transactions on 58(5):3250–3265
59. Uryas'ev S, Rubinstein RY (1994) On relaxation algorithms in computation of noncooperative equilibria. IEEE Transactions on Automatic Control 39(6):1263–1267
60. Villemonteix J, Vazquez E, Walter E (2009) An informational approach to the global optimization of expensive-to-evaluate functions. Journal of Global Optimization 44(4):509–534
61. Wagner T, Emmerich M, Deutz A, Ponweiser W (2010) On expected-improvement criteria for model-based multi-objective optimization. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 718–727
62. Wang G, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. Journal of Mechanical Design 129(4):370
63. Wilson A, Nickisch H (2015) Kernel interpolation for scalable structured gaussian processes (kiss-gp). In: International Conference on Machine Learning, pp 1775–1784
64. Žilinskas A, Zhigljavsky A (2016) Stochastic global optimization: a review on the occasion of 25 years of informatica. Informatica 27(2):229–256