Title
Adaptive grid semidefinite programming for finding optimal designs

Authors
Duarte, Belmiro P.M.
Wong, Weng Kee
Dette, Holger

Publication Date
2017-03-23

DOI
10.1007/s11222-017-9741-y

Peer reviewed

Stat Comput (2018) 28:441–460
https://doi.org/10.1007/s11222-017-9741-y

Adaptive grid semidefinite programming for finding optimal designs

Belmiro P. M. Duarte[1,2] · **Weng Kee Wong**[3] · **Holger Dette**[4]

Received: 30 July 2016 / Accepted: 15 March 2017 / Published online: 23 March 2017

Abstract We find optimal designs for linear models using a novel algorithm that iteratively combines a semidefinite programming (SDP) approach with adaptive grid techniques. The proposed algorithm is also adapted to find locally optimal designs for nonlinear models. The search space is first discretized, and SDP is applied to find the optimal design based on the initial grid. The points in the next grid set are points that maximize the dispersion function of the SDP-generated optimal design using nonlinear programming. The procedure is repeated until a user-specified stopping rule is reached. The proposed algorithm is broadly applicable, and we demonstrate its flexibility using (i) models with one or more variables and (ii) differentiable design criteria, such as A-, D-optimality, and non-differentiable criterion like E-optimality, including the mathematically more challenging case when the minimum eigenvalue of the information matrix of the optimal design has geometric multiplicity larger than 1. Our algorithm is computationally efficient because it is based on mathematical programming tools and so optimality is assured at each stage; it also exploits the convexity of the problems whenever possible. Using several linear and nonlinear models with one or more factors, we show the proposed algorithm can efficiently find optimal designs.

Keywords Adaptive grid · Continuous design · Model-based optimal design · Nonlinear programming · Semidefinite programming

Mathematics Subject Classification 62K05 · 90C47

✉ Belmiro P. M. Duarte
bduarte@isec.pt

Weng Kee Wong
wkwong@ucla.edu

Holger Dette
holger.dette@rub.de

[1] Department of Chemical and Biological Engineering, Instituto Politécnico de Coimbra, Instituto Superior de Engenharia de Coimbra, Rua Pedro Nunes, Quinta da Nora, 3030-199 Coimbra, Portugal

[2] Department of Chemical Engineering, CIEPQPF, University of Coimbra, Coimbra, Portugal

[3] Department of Biostatistics, Fielding School of Public Health, UCLA, 10833 Le Conte Ave., Los Angeles, CA 90095-1772, USA

[4] Department of Mathematics, Institute of Statistics, Ruhr-Universitaet Bochum, 44780 Bochum, Germany

1 Motivation

We consider the problem of determining model-based optimal designs of experiments for linear models as well as locally optimal designs for nonlinear models. This problem has increasing relevance in many areas, such as engineering, social sciences, food science and pharmaceutical research (Berger and Wong 2009; Goos and Jones 2011; Fedorov and Leonov 2014). The design problem is to determine optimal design points, which are members of the design space that describe the experimental condition, and the number of replicates at each of these design points, subject to the requirement that they sum to n, the maximum number of observations available for the study. These design issues involve hard combinatorial optimization problems, commonly designated the *exact design problems* that are known to be NP-hard (Welch 1982). However, since we consider *continuous* or *approximate design problems* where $n \to +\infty$, we only need to solve a P-hard reformulation of the problem to find the proportions of the optimal combination of design points. Continuous optimal designs of experiments are particularly

helpful for providing maximum information at minimum cost. For problems discussed in this paper, we assume we are given design criterion, a known *compact* design space and a known parametric linear or nonlinear model, apart from unknown parameters in the model. Typically, the goal is to find an efficient design to estimate the model parameters.

Kiefer (1959) considered that the continuous optimal design problem is equivalent to finding an optimal probability measure on the given design space $\mathbf{X}$ (Kiefer and Wolfowitz 1960; Kiefer 1974). The optimal probability measure specifies the number of design points required, where these design points are in $\mathbf{X}$ and the proportions of total observation to be taken at the design points under the design criterion (Atkinson et al. 2007). He termed these continuous designs and showed that there are many advantages of working with continuous designs (Kiefer 1959; Kiefer and Wolfowitz 1960; Kiefer 1974). Identifying optimal continuous design can be difficult to determine even for relatively simple models. Analytical solutions are rarely available for high-dimensional problems, and algorithms are required to find them, especially when the criterion is complex. The aim of this paper is to apply mathematical programming-based algorithms combined with adaptive grid techniques to find efficiently optimal continuous designs for linear and nonlinear models with one or more factors.

During the last few decades, algorithms have been developed and continually improved for generating different types of optimal designs for algebraic models. Various numerical algorithms developed for the construction of *exact designs* are based on exchange methods and were initially proposed for D-optimality criterion (Mitchell et al. 1970; Wynn 1970; Fedorov 1972). Many refinements occurred over time, among them are the DETMAX algorithm (Mitchell 1974), the modified Fedorov algorithm (Cook and Nachtsheim 1980), the KL-exchange algorithm (Atkinson and Donev 1989) and the acyclic coordinate algorithm (Meyer and Nachtsheim 1995). For *continuous optimal designs*, exchange-based algorithms for finding D-optimal designs were correspondingly developed in Wynn (1970), Wynn (1972) and Fedorov (1972), the so-called Wynn–Fedorov algorithms. Several authors contributed to improve the numerical efficiency of the Wynn–Fedorov schemes; see Wu (1978), Wu and Wynn (1978), Pronzato (2003) and Harman and Pronzato (2007). Some of these algorithms are reviewed, compared and discussed in Meyer and Nachtsheim (1995) and Pronzato (2008), among others. Multiplicative algorithm is another class of approaches to find continuous optimal designs that find broad application due to the simplicity (Mandal et al. 2015). The basic algorithm was proposed by Titterington (1976) and later exploited in Pázman (1986), Fellman (1989), Pukelsheim (1991), Torsney and Mandal (2006), Mandal and Torsney (2006), Dette (2008), Torsney and Martín-Martín (2009), Yu (2010c, 2010b). Recently, the cocktail algorithms that rely on both exchange and multiplicative algorithms have been proposed (Yu 2010a) and improved (Yang et al. 2013).

The algorithms are iterative, requiring a starting design and a stopping criterion to search for the optimal solution. The stopping criterion may be the maximum number of iterations allowed or the requirement that the value of the optimality criterion of the generated design does not change from the previous values by some pre-specified tolerance level. The algorithms iteratively replace current design points by one or more points that are new or already in the support of the current design. The rule for selecting the point or points for generating the next design varies depending on the type of algorithms and the design criterion. Some issues of such algorithms are the need to collapse points very close together to a support point of the design, and how often this procedure needs to be carried out.

Mathematical programming algorithms have improved substantially over the last two decades, and they can currently solve complex high-dimensional optimization problems, especially when they are P-hard. Examples of applications of mathematical programming algorithms for finding continuous optimal designs are linear programming (Gaivoronski 1986; Harman and Jurík 2008), second-order conic programming (Sagnol 2011; Sagnol and Harman 2015), semidefinite programming (SDP) (Vandenberghe and Boyd 1999; Papp 2012; Duarte and Wong 2015), semi-infinite programming (SIP) (Duarte and Wong 2014; Duarte et al. 2015) and nonlinear programming (NLP) (Chaloner and Larntz 1989; Molchanov and Zuyev 2002). Applications based on optimization procedures relying on metaheuristic algorithms are also reported in the literature; see Heredia-Langner (2004) for genetic algorithms, Woods (2010) for simulating annealing, Chen et al. (2015) for particle swarm optimization and Masoudi et al. (in press) for imperialist competitive algorithm, among others.

In this paper, we focus on SDP, which is not new; details on the general use and application of SDP to search for optimal designs for linear models are available in Vandenberghe and Boyd (1996). Additional applications include finding (i) D-optimal designs for multi-response linear models (Filová et al. 2011), (ii) c-optimal designs for single-response trigonometric regression models (Qi 2011), (iii) D-optimal designs for polynomial models and rational functions (Papp 2012) and (iv) Bayesian optimal designs for nonlinear models (Duarte and Wong 2015). A key advantage of using SDP to handle the design problem is that it transforms the original problem into a convex program that allows us to efficiently find the global optimal design. However, drawbacks are that (i) the design space has to be discretized and consequently this may produce sub-optimal designs when the design space is continuous and the grid is coarse, and (ii) the success of

the strategy depends on the dimension of the problem and the types of SDP solvers available.

A potential strategy to circumvent the drawbacks of SDP is to use adaptive grid (AG) strategies where the grid used to search for the optimal design can be increasingly reduced in size and locations of the support points can be more accurately located at the same time. As we will show, having an adaptive grid search with a coarse initial grid also does not seem to have an impact on the computational time and quality of the optimal design generated. Grid adaptation search strategy is commonly employed to solve PDEs and computational fluid dynamics problems where it is important that "eventual moving fronts are well followed by meshes not much dense" (Berger 1982; Peraire et al. 1987). The rationale of the adaptive grid search for the optimal support points is similar to the step of deletion/exchange of points in the several exchange algorithms (Atkinson et al. 2007, Chap. 12) previously used in the literature. After the initial user-specified grid used to find the optimal design, the next grid is generated by points that maximize a specific function formed from the current design. The steps are repeated until a user-specified rule for convergence is met. Unlike previously proposed algorithms, such as Fedorov's algorithm where only one point is allowed to augment the current design in each iteration, our method has the advantages of (i) working with only points that maximize the directional derivative of the criterion evaluated at the SDP-generated design and (ii) the subsequent grid sets can be substantially smaller than initial grid set so that the optimization problem to find the support points of the optimal design is increasingly simplified by having to search over a few candidate points.

AG approaches have never been combined with mathematical programming formulations to find optimal designs, and an exception is Pronzato and Zhigljavsky (2014, Sect. 3.4). In our proposed methodology, we have two levels of optimization: (i) the SDP solver finds the optimal design for a given grid and (ii) the AG algorithm finds a new grid (node's placement) that consists of points that maximize the directional derivative of the criterion evaluated at the current design. The later procedure requires solving a constrained nonlinear program. We present an algorithm that automates the process and tests it for linear and nonlinear algebraic models.

Section 2 presents the statistical setup, a brief review of optimal design theory, and how to verify whether a design is optimal or not. Section 3 describes the mathematical formulations and algorithmic procedure used to find optimal designs by updating the grid judiciously. Section 4 applies our algorithm to find different types of optimal designs for various linear and nonlinear models with one or more variables. We offer a summary in Section 5.

2 Background

In this section, we provide the background material required for the formulation and numerical solution of optimal experimental design problems. In Sect. 2.1, we introduce SDP as a tool to find optimal designs, and in Sect. 2.2, we briefly review the fundamentals of NLP.

Throughout we assume we have a nonlinear model with a given differentiable mean function $f(\boldsymbol{x}, \boldsymbol{\beta}^\top)$ with independent components and $\boldsymbol{x} \in \mathbf{X} = \bigotimes_{i=1}^{n_x} [x_i^{LO},\ x_i^{UP}] \subset \mathbb{R}^{n_x}$. The design space $\mathbf{X}$ is typically the Cartesian product of the domains of the variables and has dimension n_x and each x_i in $\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_x})$ has a known range given by its lower bound x_i^{LO} and its upper bound x_i^{UP}. The univariate response is $y \in \mathbb{R}$, and its mean response at $\boldsymbol{x}$ is modeled by

$$\mathbb{E}[y|\boldsymbol{x}, \boldsymbol{\beta}] = f(\boldsymbol{x}, \boldsymbol{\beta}), \tag{1}$$

where the vector of unknown model parameters is $\boldsymbol{\beta} \in \mathbf{P}$, a known n_p-dimensional Cartesian box $\mathbf{P} \equiv \times_{j=1}^{n_p} [l_j,\ u_j]$, with each interval $[l_j,\ u_j]$ representing the plausible range of values for the jth parameter. The symbol $\mathbb{E}[\bullet]$ is the expectation operator, and the linear model is a subclass of (1) where $\mathbb{E}[y|\boldsymbol{x}, \boldsymbol{\beta}] = \boldsymbol{\beta}^\top\ f(\boldsymbol{x})$. When nonlinear models are considered, we assume that a nominal set of values of the parameters are known *a priori* and our goal is to determine locally optimal designs that can be used to confirm/improve their accuracy. Finally, we assume that the errors of the response are independent and homoscedastic.

Suppose we have a continuous design with $k(\leq n)$ support points at $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ and the weights at these points are, respectively, $w_1, w_2, \ldots, w_k$. To implement the design for a total of n observations, we take roughly $n \times w_i$ observations at $\boldsymbol{x}_i, i = 1, \ldots, k$ subject to $n \times w_1 + \cdots + n \times w_k = n$ and each summand is an integer. If there are n_x variables in the model, we denote the ith support point by $\boldsymbol{x}_i^\top = (x_{i,1}, \ldots, x_{i,n_x})$ and represent the design ξ by k rows $(\boldsymbol{x}_i^\top, w_i),\ i \in \{1, \ldots, k\}$ with $\sum_{i=1}^k w_i = 1$. In what is to follow, we let $\Xi \equiv \mathbf{X}^k \times \Sigma$ be the space of feasible k-point designs over $\mathbf{X}$ where Σ is the $k-1$-simplex in the domain of weights $\Sigma = \{\sum_{i=1}^k w_i :\ w_i \geq 0,\ \forall i \in [k],\ \sum_{i=1}^k w_i = 1\}$, and let $[k] = \{1, \ldots, k\}$.

Following convention, we measure the worth of a design by its Fisher Information Matrix (FIM). The elements of the normalized FIM are the negative expectation of the second-order derivatives of the log-likelihood of (1), $\mathscr{L}(\xi, \boldsymbol{\beta})$, with respect to the parameters, given by

$$\begin{aligned} \mathscr{M}(\xi) &= -\mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\beta}}\left(\frac{\partial \mathscr{L}(\xi)}{\partial \boldsymbol{\beta}^\top}\right)\right] \\ &= \int_{\xi \in \Xi} M(\boldsymbol{x})\ \mathrm{d}(\xi) = \sum_{i=1}^k w_i\ M(\boldsymbol{x}_i), \end{aligned} \tag{2}$$

where $\mathscr{M}(\xi)$ is the *global* FIM from the design ξ, $M(\boldsymbol{x}_i)$ is the *local* FIM from point $\boldsymbol{x}_i$. Here and throughout, we use bold face lowercase letters to represent vectors, bold face capital letters for continuous domains, blackboard bold capital letters for discrete domains, and capital letters for matrices. For example, let $\mathbb{X}$ be the discretized version of $\mathbf{X}$ with, say, q points and let $[q] = \{1, \ldots, q\}$ be the set containing the point's identification. Without loss of generality, we assume that each covariate space, a subspace of the design space $\mathbf{X}$, is discretized by uniformly spaced points with possibly different step sizes (Δx_i, $\forall i$) for the different covariate spaces. The integral in (2) may be represented by

$$\mathscr{M}(\xi) = \sum_{\boldsymbol{x} \in \mathbb{X}} M(\boldsymbol{x})\ \chi(\boldsymbol{x}) \tag{3}$$

where χ is the continuous design with the same support points and weight distribution on $\mathbb{X}$.

We focus on the class of design criteria proposed by Kiefer (1974). Each member in the class is indexed by a parameter δ, is positively homogeneous and is defined on the set of symmetric $n_p \times n_p$ semi-positive definite matrices given by

$$\Phi_\delta[\mathscr{M}(\xi)] = \left[\frac{1}{n_p}\ \mathrm{tr}(\mathscr{M}(\xi)^\delta)\right]^{1/\delta}. \tag{4}$$

The maximization of Φ_δ for $\delta \neq 0$ is equivalent to minimization of $\mathrm{tr}(\mathscr{M}(\xi)^\delta)$ when $\delta < 0$. We note that Φ_δ becomes $[\mathrm{tr}(\mathscr{M}(\xi)^{-1})]^{-1}$ for $\delta = -1$, which is A-optimality, and becomes $\lambda_{\min}[\mathscr{M}(\xi)]$ when $\delta = -\infty$, which is E-optimality, and $[\det[\mathscr{M}(\xi)]]^{1/n_p}$ when $\delta \to 0$, which is D-optimality. These design criteria are suitable for estimating model parameters as they maximize the FIM in various ways. For example, when standard normal linear models and the D-optimality criterion are considered, the volume of the confidence region of $\boldsymbol{\beta}$ is proportional to $\det[\mathscr{M}^{-1/2}(\xi)]$, and consequently maximizing the determinant or a convenient function of the determinant (e.g., logarithm or geometric mean) of the FIM leads to the smallest possible volume.

When the design criterion is convex or concave (which is the case for the above criteria), the global optimality of a design ξ in $\mathbf{X}$ can be verified using an equivalence theorem based on directional derivative considerations (Kiefer and Wolfowitz 1960; Fedorov 1972; Whittle 1973; Kiefer 1974; Silvey 1980; Pukelsheim 1993). For instance, if we let $\boldsymbol{\delta}_x$ be the degenerate design at the point $\boldsymbol{x} \in \mathbf{X}$, the equivalence theorems for D-, A- and E-optimality are as follows: (i) ξ_D is D-optimal if and only if

$$\mathrm{tr}\left\{[\mathscr{M}(\xi_D)]^{-1}\ M(\boldsymbol{\delta}_x)\right\} - n_p \leq 0, \quad \forall \boldsymbol{x} \in \mathbf{X}; \tag{5}$$

(ii) ξ_A is globally A-optimal if and only if

$$\mathrm{tr}\left\{[\mathscr{M}(\xi_A)]^{-2}\ M(\boldsymbol{\delta}_x)\right\} - \mathrm{tr}\left\{[\mathscr{M}(\xi_A)]^{-1}\right\} \leq 0, \quad \forall \boldsymbol{x} \in \mathbf{X}, \tag{6}$$

and (iii) ξ_E is globally E-optimal if and only if (Dette and Studden 1993)

$$\min_{\mathbf{E} \in \mathscr{E}}\ \mathrm{tr}\{\mathbf{E}\ M(\boldsymbol{\delta}_x)\} - \lambda_{\min} \leq 0, \quad \forall \boldsymbol{x} \in \mathbf{X}, \tag{7}$$

where $\mathscr{E}$ is the space of $n_p \times n_p$ positive semidefinite matrices with trace equal to 1 and $\mathbf{E} \in \mathscr{E}$ has the form

$$\mathbf{E} = \sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},\ i}\ \boldsymbol{e}_{\lambda_{\min},\ i}^{\mathsf{T}}\right). \tag{8}$$

Here $\boldsymbol{e}_{\lambda_{\min},1}, \ldots, \boldsymbol{e}_{\lambda_{\min},m_\lambda}$ are normalized linearly independent eigenvectors of $\mathscr{M}(\xi_E)$ corresponding to $\lambda_{\min}$ with geometric multiplicity m_λ and $\alpha_1, \ldots, \alpha_{m_\lambda}$ are nonnegative weights that sum to unity. We recall the geometric multiplicity, or simply the multiplicity of an eigenvalue is the number of linearly independent eigenvectors associated with the eigenvalue.

We call the functions on the left side of the inequalities (5, 6 and 7) *dispersion functions* and denote them by $\Psi(\boldsymbol{x}|\xi)$. They are different for different concave criteria.

2.1 Semidefinite programming

Semidefinite programming is employed to solve the optimal design problems for D-, A- and E-optimality criteria over a given discrete domain $\mathbb{X}$. In this section, we introduce the fundamentals of this class of mathematical programs.

Let $\mathbb{S}^{n_p}$ be the space of $n_p \times n_p$ symmetric semidefinite positive matrices. A function $\varphi\colon \mathbb{R}^{m_1} \mapsto \mathbb{R}$ is called semidefinite representable (SDr) if and only if inequalities of the form $u \leq \varphi(\boldsymbol{\zeta})$, where $\boldsymbol{\zeta} \in \mathbb{R}^{m_1}$ is a vector, can be expressed by *linear matrix inequalities* (LMI) (Ben-Tal and Nemirovski 2001; Boyd and Vandenberghe 2004). That is, $\varphi(\boldsymbol{\zeta})$ is SDr if and only if there exist some symmetric matrices $M_0, \ldots, M_{m_1}, \ldots, M_{m_1+m_2} \in \mathbb{S}^{n_p}$ such that

$$u \leq \varphi(\boldsymbol{\zeta}) \iff \exists \boldsymbol{v} \in \mathbb{R}^{m_2} :\ u\ M_0 + \sum_{i=1}^{m_1} \zeta_i\ M_i + \sum_{j=1}^{m_2} v_j\ M_{m_1+j} \succeq 0. \tag{9}$$

Here, $\succeq$ is the semidefinite operator, i.e., $A \succeq 0 \iff \langle A\,\boldsymbol{\zeta}, \boldsymbol{\zeta}\rangle > 0$, $\forall \boldsymbol{\zeta} \in \mathscr{H}$, where $\langle ., . \rangle$ is the Frobenius inner product operator and $\mathscr{H}$ is the Hilbert space. The optimal values, $\boldsymbol{\zeta}$, of SDr functions are then formulated as *semidefinite programs* of the form:

$$\max_{\boldsymbol{\zeta}} \left\{ \boldsymbol{c}^{\mathsf{T}}\, \boldsymbol{\zeta}, \sum_{i=1}^{m_1} \zeta_i\, M_i - M_0 \succeq 0 \right\} \tag{10}$$

In our design context, $\boldsymbol{c}$ is a vector of known constants that depends on the design problem, and matrices $M_i,\ i = \{0, \ldots, m_1\}$ contain local FIM's and other matrices produced by the reformulation of the functions $\varphi(\boldsymbol{\zeta})$. The decision variables in vector $\boldsymbol{\zeta}$ are the weights $w_i,\ i \in [q]$ of the optimal design and other auxiliary variables required. The problem of calculating a design for a pre-specified grid $\mathscr{G}$ of points $\boldsymbol{x}_i$ is solved with the formulation (10) complemented with the linear constraints on $\boldsymbol{w}$: (i) $\boldsymbol{w} \succeq 0$, and (ii) $\mathbf{1}^{\mathsf{T}}\, \boldsymbol{w} = 1$.

Ben-Tal and Nemirovski (2001, Chap. 2, 3) provided a list of SDr functions in SDP formulations useful for solving continuous optimal design problems; see Boyd and Vandenberghe (2004, Sect. 7.3). Sagnol (2013) showed that each criterion in the Kiefer's class of optimality criteria defined by (4) is SDr for all rational values of $\delta \in (-\infty, -1]$ and general SDP formulations exist. This result also applies to the case when $\delta \to 0$.

2.2 Nonlinear programming

In this section, we introduce NLP which is used to find the points that maximize the dispersion function over the continuous design domain. Nonlinear programming seeks to find the global optimum $\boldsymbol{x}$ of a convex or nonconvex nonlinear function $f : \mathbf{X} \mapsto \mathbb{R}$ in a compact domain $\mathbf{X}$ with possibly nonlinear constraints. The general structure of the NLP problems is:

$$\min_{\boldsymbol{x} \in \mathbf{X}} \ f(\boldsymbol{x}) \tag{11a}$$

$$\text{s.t. } \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \tag{11b}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \tag{11c}$$

where (11b) represents a set of r_i inequalities, and (11b) represents a set of r_e equality constraints. The functions $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are twice differentiable, and in our context, the variable $\boldsymbol{x} \in \mathbf{X}$ contains points that we want to choose from the candidate set of maximizers of the dispersion function $\Psi(\boldsymbol{x}|\xi)$, and $f(\boldsymbol{x})$ is a convex linear combination of $\Psi(x|\xi)$ for a pre-specified k-point design obtained with SDP. The variables are subject to bounds previously set based on the SDP-generated design, and this topic is further discussed in Sect. 3.2.

Nested and gradient projection methods are commonly used to solve NLP problems, and some examples are the general reduced gradient (GRG) (Drud 1985, 1994) and trust region (Coleman and Li 1994) algorithms. Other methods are sequential quadratic programming (SQP) (Gill et al. 2005) and interior point (IP) (Byrd et al. 1999). Ruszczyński (2006) provides an overview of NLP algorithms.

3 Algorithm

This section describes an algorithm for finding D-, A- and E-optimal designs for linear models employing an SDP-based procedure combined with AG. In Sect. 3.1, we introduce the formulation to find SDP-based designs, and in Sect. 3.2, we discuss the adaptive grid algorithm. Because E-optimality is not differentiable, the grid adaptation procedure has to be modified, particularly so when the problem has multiple minimum eigenvalues; Sect. 3.3 describes the strategy for this case.

3.1 Semidefinite programming formulation

The SDP formulations for finding optimal designs for linear models are based on the representations of Boyd and Vandenberghe (2004), and require a pre-specified grid $\mathscr{G} = \{\boldsymbol{x} : x_1 \leq x_2 \leq \cdots \leq x_{q-1} \leq x_q\}$ of points over $\mathbf{X}$. The global FIM is constructed by averaging the local FIM's, and the SDP solver determines the weights at each point so that the design optimality criterion as a function of the FIM is optimized. The solver determines automatically the number of support point of the SDP-generated design, and these are from points in the current grid with positive weights.

The SDP formulation for solving the D-optimal design problem can be more compactly represented by

$$z = \max_{\boldsymbol{w} \in \mathbb{R}^q} [\det(\mathscr{M}(\xi))]^{1/n_p} \tag{12a}$$

$$\text{s.t. } \sum_{i=1}^{q} w_i = 1 \tag{12b}$$

$$\mathscr{M}(\xi) \succeq 0 \tag{12c}$$

$$w_i \geq 0, \quad \forall i \in [q], \tag{12d}$$

which can then be transformed into LMIs and solved with a SDP solver. As an illustration, suppose we specialize this general approach to find a D-optimal design. We recall that the LMI $\tau \leq (\det[\mathscr{M}(\xi)])^{1/n_p}$ holds if and only if there exists a $n_p \times n_p$-lower triangular matrix $\mathscr{C}$ such that

$$\begin{bmatrix} \mathscr{M}(\xi) & \mathscr{C}^{\mathrm{T}} \\ \mathscr{C} & \operatorname{diag}(\mathscr{C}) \end{bmatrix} \succeq 0 \quad \text{and} \quad \tau \leq \left(\prod_{j=1}^{n_p} \mathscr{C}_{j,j} \right)^{1/n_p},$$

where $\operatorname{diag}(\mathscr{C})$ is the diagonal matrix with diagonal entries $\mathscr{C}_{j,j}$ and the geometric mean of the $\mathscr{C}_{j,j}$'s can, in turn, be expressed as a series of 2×2 LMIs (Ben-Tal and Nemirovski 2001).

The formulations for finding A- and E-optimal designs are given below in (13) and (14), respectively:

$$z = \max_{\boldsymbol{w} \in \mathbb{R}^q} [\operatorname{tr}(\mathscr{M}^{-1}(\xi))]^{-1} \tag{13a}$$

$$\text{s.t.} \sum_{i=1}^{q} w_i = 1 \tag{13b}$$

$$\mathscr{M}(\xi) \succeq 0 \tag{13c}$$

$$w_i \geq 0, \quad \forall i \in [q] \tag{13d}$$

$$z = \max_{\boldsymbol{w} \in \mathbb{R}^q} [\lambda_{\min}(\mathscr{M}(\xi))] \tag{14a}$$

$$\text{s.t.} \sum_{i=1}^{q} w_i = 1 \tag{14b}$$

$$\mathscr{M}(\xi) \succeq 0 \tag{14c}$$

$$w_i \geq 0, \quad \forall i \in [q]. \tag{14d}$$

We denote the design problems (12, 13 and 14) by $\mathscr{P}_1$ and employ a user-friendly interface, `cvx` (Grant et al. 2012), to solve them. The `cvx` environment automatically transforms the constraints of the form $\tau \leq \varphi(\boldsymbol{\zeta})$ into a series of LMIs, which are then passed on to SDP solvers such as `SeDuMi` (Sturm 1999) or `Mosek` (Andersen et al. 2009). All the results presented in Sect. 4 were obtained with `Mosek`.

3.2 Adaptive grid procedure

This section describes the procedure to adaptively refine the grid and delete candidate nodes when they are not required in the design. We assume the search space is one-dimensional, i.e., $\mathbf{X} \in \mathbb{R}$ and discuss the extension to $n_x \geq 2$ later on.

We begin the procedure with an equidistributed grid $\mathscr{G}^{(0)}$, where the superscript indicates the iteration number, so at iteration j, the grid set becomes $\mathscr{G}^{(j)}$. The value of Δx is self-selected to be automatically computed from the number of candidate points q in $\mathbf{X}$. For this grid set, we solve the problem of interest in $\mathscr{P}_1$ and denote the SDP optimal design and the criterion value by $\xi^{(0)}$ and $z^{(0)}$, respectively.

Suppose $\xi^{(0)}$ has $k^{(0)}$ support points and its dispersion function is $\Psi(\boldsymbol{x}|\xi^{(0)})$, $\boldsymbol{x} \in \mathbf{X}$. Points in $\mathbf{X}$ that maximize this dispersion function become points in the new grid set forming $\mathscr{G}^{(1)}$; this is accomplished by solving a constrained NLP problem. To distinguish the support points of the design $\xi^{(0)}$ from the candidate points in the updated grid $\mathscr{G}^{(1)}$, we designate the former by $\boldsymbol{s}_i^{(0)}$, $i \in [k^{(0)}]$, and the latter set by $\boldsymbol{x}_i^{(1)}$, $i \in [k^{(0)}]$. The grid $\mathscr{G}^{(1)}$ has up to $k^{(0)}$ points and is determined by solving the problem (15). In practice, the grid $\mathscr{G}^{(1)}$ has $k^{(0)}$ potential points, each one corresponding to a local maximizer of the dispersion function, but may be fewer if some of them are the same or very close, in which case, they are collapsed to a single point. The optimization problem is:

$$\max_{\boldsymbol{x}^{(1)} \in \mathbf{X}} \frac{1}{k^{(0)}} \sum_{i=1}^{k^{(0)}} \Psi(\boldsymbol{x}_i^{(1)}|\xi^{(0)}) \tag{15a}$$

$$\text{s.t } \boldsymbol{x}_i^{(1)} \geq \boldsymbol{s}_i^{(0)} - \Delta\boldsymbol{x}, \quad \boldsymbol{x}_i^{(1)} \leq \boldsymbol{s}_{i+1}^{(0)} - \Delta\boldsymbol{x}, \quad i \in \{2, \cdots, k^{(0)} - 1\} \tag{15b}$$

$$\boldsymbol{x}_1^{(1)} \geq \boldsymbol{x}^{LO}, \quad \boldsymbol{x}_i^{(1)} \leq \boldsymbol{s}_2^{(0)} - \Delta\boldsymbol{x} \tag{15c}$$

$$\boldsymbol{x}_{k^{(0)}}^{(1)} \leq \boldsymbol{x}^{UP}, \quad \boldsymbol{x}_i^{(1)} \geq \boldsymbol{s}_{k^{(0)}-1}^{(0)} - \Delta\boldsymbol{x} \tag{15d}$$

where (15a) is the objective function and (15b, 15c and 15d) are bound constraints for each maximizer, where the bounds are constructed from the support points of the previous SDP optimal design as described below. We designate the problem (15) as $\mathscr{P}_2$, and its formalization is based on two interesting features. First, the dispersion function is often nonconvex and finding a single maximum is a challenging task. Second, each maximizer is independent on the others. Let us first introduce the rational used to construct bounds for local maximizers, and subsequently present the procedure to collapse maximizers, when required.

The design points obtained from the SDP provide initial estimates of the maximizers of the dispersion function and are also used to construct bounds for them. Let one of the design points obtained by solving the SDP problem at iteration 0 be $\boldsymbol{s}_i^{(0)}$, and another be $\boldsymbol{s}_{i+1}^{(0)}$. Then there exists a local maximizer of the dispersion function, $\boldsymbol{x}_i^{(1)}$, in the interval $\underline{\mathbf{X}}_i \equiv [\boldsymbol{s}_i^{(0)} - \Delta\boldsymbol{x},\ \boldsymbol{s}_{i+1}^{(0)} - \Delta\boldsymbol{x}] \in \mathbf{X}$, corresponding to the design space between the candidate points $\boldsymbol{s}_{i-1}^{(0)}$ and $\boldsymbol{s}_{i+1}^{(0)}$. We note that the problem of maximizing the dispersion function in interval $\underline{\mathbf{X}}_i$ is convex and the optimum found is global. We apply this procedure to all the support points of the design $\xi^{(0)}$ and construct non-overlapping bounding intervals $\underline{\mathbf{X}}_i$ for each local maximizer of $\Psi(\boldsymbol{x}|\xi^{(0)})$. This strategy requires solving a different NLP problem for each compact interval $\underline{\mathbf{X}}_i$. However, since all maximizers are independent of each other and by construction they have equal values of $\Psi(\boldsymbol{x}|\xi^{(0)})$, we formulate the objective function of the resulting NLP problem as a linear combination of the values of the dispersion function from all the compact intervals $\underline{\mathbf{X}}_i$, $\forall i \in [k^{(0)}]$. Since each $\boldsymbol{x}_i^{(1)}$ is locally constrained by (15b), we can solve the problem using a single call of the solver. Further, because the problem is convex and the solver guarantees globally optimal solutions, all of the maximizers are located. It is worth noting that the constraints (15b) of the NLP problem have a tridiagonal structure that is exploited by decomposition techniques to increase the numerical efficiency of the solver.

To check whether contiguous maximizers need to be collapsed, we first measure the distance between successive candidate points in $\boldsymbol{x}^{(1)}$ by

$$d\left(x_{i+1}^{(1)}, x_i^{(1)}\right) = \left\| x_{i+1}^{(1)} - x_i^{(1)} \right\|_2, \quad i \in [k^{(0)} - 1]. \tag{16}$$

When two points are ϵ-close for a pre-defined ϵ, i.e., $d(x_{i+1}^{(1)}, x_i^{(1)}) < \epsilon$, the points are collapsed into a single point with average values of $\boldsymbol{x}$ included in the new grid $\mathscr{G}^{(1)}$; otherwise, both points are included. After collapsing two points, the criterion measuring the design efficiency does not decrease because this procedure is only performed when two local maximizers are within ϵ-close in $\mathbb{R}^{n_x}$. Further, the local maximizers of the dispersion function in iteration $j+1$ are at least as efficient as that obtained from the SDP-generated design from the previous iteration because the NLP problem finds the local maxima of the directional derivative on a set of bounded compact subintervals for each $\boldsymbol{x}_i^{(1)}$. In the worst case, the maximizers are equal to previous support points and the efficiency of the design neither increases nor decreases which happens when convergence is attained.

The grid $\mathscr{G}^{(1)}$ replaces $\mathscr{G}^{(0)}$, and the optimal design $\xi^{(1)}$ for this new grid is obtained with the SDP formulation. The optimum is saved as $z^{(1)}$, and the procedure terminates if the following convergence criterion (17) is met:

$$\left| \frac{z^{(j)} - z^{(j-1)}}{z^{(j)}} \right| \leq \epsilon_1 \tag{17}$$

The value of the relative tolerance ϵ_1 is also user specified. If the condition (17) is not satisfied, the procedure is repeated, starting with the solution of $\mathscr{P}_2$ for the dispersion function obtained from $k^{(1)}$-support points design $\xi^{(1)}$. In every iteration, the NLP problem (15) is solved with an interior point-based solver, `IPOPT` (Wächter and Biegler 2005). To increase the accuracy, the gradient and Jacobian matrix required by the solver are constructed employing an automatic differentiation tool, `ADiMat` (Bischof et al. 2002).

Algorithm 1 below summarizes the procedure. The distinguishing feature of the proposed algorithm is that it converges to the global optimal design, ξ^*. To see this, let us consider that in $(j-1)$th iteration we have the design $\xi^{(j-1)}$ with the support points $\boldsymbol{s}^{(j-1)}$. The maximizers are $\boldsymbol{x}^{(j)} = \max_{\boldsymbol{x}} \Psi(\boldsymbol{x}|\xi^{(j-1)})$ and consequently

$$\Psi\left(\boldsymbol{x}^{(j)}|\xi^{(j-1)}\right) - \Psi\left(\boldsymbol{s}^{(j-1)}|\xi^{(j-1)}\right) \geq 0. \tag{18}$$

Then, a new grid is formed with the candidate points $\boldsymbol{s}^{(j)}$ that are the local maximizers $\boldsymbol{x}^{(j)}$, and a new optimal design $\xi^{(j)} = \max_{\xi} \Phi_\delta(\mathscr{M}(\xi))$ is found for this new grid. Using the relation (18), $\Phi_\delta(\mathscr{M}(\xi^{(j)})) \geq \Phi_\delta(\mathscr{M}(\xi^{(j-1)}))$ and since the dispersion function is bounded from above, it follows that $\Phi_\delta(\mathscr{M}(\xi))$ converges globally and tends to a finite value.

We also compare our algorithm with traditional exchange and multiplicative algorithms. The improvement of the optimality criterion in exchange algorithms is performed by updating the weight and the location of each of the candidate points in the design space, and both steps are sequential. The multiplicative algorithm allows us to update simultaneously the weight and the location of each candidate point. Here, we start with a pre-defined grid before updating the weights of all the candidate points simultaneously which is performed by solving the SDP problem, whereupon the complete grid of candidate points is also updated simultaneously by solving the NLP problem. We use a collapsing scheme to join local maximizers ϵ-close, similarly to the other algorithms that consider collapsing and acceleration procedures. We expect to converge the design in few iterations, since both updating steps are global and not local as in the other algorithms. However, every step in our algorithm is more complex and requires global solvers so that the convergence is assured. Computationally, the more demanding step in our algorithm is the first SDP problem where all the candidate points are considered. The NLP problems to solve are of small dimension, and the SDP problems in subsequent iterations are also soft, since the number of candidate points is reduced.

All computations in this paper were carried using on an Intel Core i7 machine (Intel Corporation, Santa Clara, CA) running 64 bits Windows 10 operating system with 2.80 GHz. The relative and absolute tolerances used to solve the SDP and NLP problems were set to 10^{-5}. The values of ϵ and ϵ_1 in (16) and (17), respectively, are also set to 10^{-5} for all the problems addressed.

Algorithm 1 Algorithm to find optimal designs combining SDP with AG.

procedure OPTIMALDESIGN(x^L, x^U, q, ϵ, criterion)
  $\Delta x \leftarrow (x^U - x^L)/(q-1)$ ▷ Compute the disc. interval
  $j \leftarrow 0$ ▷ Initialize the it. counter
  Construct $\mathscr{G}^{(j)}$ using intervals Δx ▷ Discretization of the design space
  Find $\xi^{(j)}$ ▷ Solve SDP problem $\mathscr{P}_1$
  $z^{(j)} \leftarrow z$
  Find new candidate points ▷ Solve NLP problem $\mathscr{P}_2$
  Check points distance using (16) ▷ Collapse points if needed
  $j \leftarrow j+1$
  $\mathscr{G}^{(j)} \leftarrow \boldsymbol{x}^{(j-1)}$ ▷ Update the grid
  Find $\xi^{(j)}$ ▷ Solve SDP problem $\mathscr{P}_1$
  $z^{(j)} \leftarrow z$
  while $|(z^{(j)} - z^{(j-1)})/z^{(j)}| > \epsilon$ **do** ▷ Convergence checking
    Find new candidate points ▷ Solve NLP problem $\mathscr{P}_2$
    Check points distance using (16) ▷ Collapse points if needed
    $j \leftarrow j+1$
    $\mathscr{G}^{(j)} \leftarrow \boldsymbol{x}^{(j-1)}$ ▷ Update the grid
    Find $\xi^{(j)}$ ▷ Solve SDP problem $\mathscr{P}_1$
    $z^{(j)} \leftarrow z$
  end while
end procedure

3.3 Adaptive strategy for finding E-optimal designs

Section 3.2 applies the adaptive grid strategy to construct D- and A- designs. The methodology can also be extended

to E-optimality, which is a non-differentiable criterion. For E-optimality, we focus on the minimum eigenvalue of the information matrix and consider separately, the simpler case when its geometric multiplicity is $m_\lambda = 1$ and the more difficult case when it is larger than 1. When the multiplicity of the minimum eigenvalue, m_λ, is 1, there is only one nonzero α in (8), resulting in a simple dispersion function to maximize, and Algorithm 1 can be used without modification.

The case with $m_\lambda \geq 2$ occurs in applications, such as in one-dimensional polynomial models with "large" design spaces (Melas 2006), or in studying response surface models (Dette and Grigoriev 2014). When $m_\lambda \geq 2$, it is harder to verify condition (7) because we now have to additionally determine the weights $\alpha_1, \ldots, \alpha_{m_\lambda}$. These weights play a crucial role because (i) failure to determine the weights correctly may lead us to continue search for the optimal design even when the current design is optimal and (ii) the computational time to find the optimal design depends on how fast these weights are identified correctly. The upshot is that maximizing the dispersion function of the SDP-generated design using NLP becomes more challenging.

We recall that given an initial grid $\mathscr{G}^{(0)}$, we first use SDP to find a $k^{(0)}$-point optimal design and use its dispersion function to ascertain whether it satisfies the conditions in Sect. 2, cf. (7). Let $\lambda_{\min}^{(0)}$ be the minimum eigenvalue of $\mathscr{M}(\xi_E^{(0)})$, m_λ be its multiplicity, and $\mathscr{S}^{(0)} = \boldsymbol{s}_l,\ l \in [k^{(0)}]$ be the set of support points of the design $\xi_E^{(0)}$. First, we determine the optimal combination of $\boldsymbol{\alpha}$ that minimizes the mean absolute deviation of the dispersion function at the support points $\boldsymbol{s}_l,\ l \in [k^{(0)}]$ of $\xi_E^{(0)}$. This task is carried out by solving the following constrained linear programming (LP) problem similar to Arthanari and Dodge (1993, Chap. 2):

$$\min_{t,\boldsymbol{\alpha}} \sum_{l=1}^{k^{(0)}} t_l \tag{19a}$$

$$\text{s.t. } \operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\mathsf{T}}\right) M(\delta_{\boldsymbol{s}_l})\right] - \lambda_{\min}^{(0)} \leq t_l, \quad \boldsymbol{s}_l \in \mathscr{S}^{(0)} \tag{19b}$$

$$\operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\mathsf{T}}\right) M(\delta_{\boldsymbol{s}_l})\right] - \lambda_{\min}^{(0)} \geq -t_l, \quad \boldsymbol{s}_l \in \mathscr{S}^{(0)} \tag{19c}$$

$$\operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\mathsf{T}}\right) M(\delta_{\boldsymbol{x}_j})\right] - \lambda_{\min}^{(0)} \leq 0, \quad \boldsymbol{x}_j \in \mathscr{G}^{(0)} \setminus \mathscr{S}^{(0)} \tag{19d}$$

$$\sum_{i=1}^{m_\lambda} \alpha_i = 1, \tag{19e}$$

$$\text{and } t_l \geq 0. \tag{19f}$$

Here (19b) and (19c) represent the upper and lower bounds of the error of the dispersion function at the support points, respectively, and $\boldsymbol{e}_{\lambda_{\min},i}$ is the eigenvector associated with ith smallest eigenvalue of $\mathscr{M}(\xi_E^{(0)})$. Equation (19d) guarantees that the fitted dispersion function is below $\lambda_{\min}^{(0)}$ for all points from the initial grid except the support points. Since the problem (19) falls into LP class and the number of decision variables, $m_\lambda + k^{(0)}$, is small, very little computational effort is required to find the global optimum. We also use `Mosek` to handle (19) using a tolerance level of 10^{-5}.

The next step in the extended algorithm computes the matrix $\mathbf{E}$ from $\boldsymbol{\alpha}$ using Eq. (8) and solves problem (15) by finding points $\boldsymbol{x}^{(1)}$ that maximize the dispersion function. From this point on, the extended algorithm runs the same way it did in Sect. 3.2 for $m_\lambda = 1$. If Algorithm 1 requires a few iterations to converge, problem (19) is solved at every iteration after replacing $\mathscr{G}^{(0)}$ by the latest grid, and replacing the $k^{(0)}$-point design $\xi_E^{(0)}$ by the SDP-generated design obtained with the latest grid.

4 Applications to find D-, A- and E-optimal designs

We apply our algorithms in Sect. 3 to find D-, A- and E-optimal designs for a battery of linear and nonlinear models in Table 1. For models 1–4, the design space is $\mathbf{X} = [-1, 1]$ and for Models 5–8 other design spaces are used, some of them replicating the original problem. Models 1–5 are linear and Models 6–8 are nonlinear and we are interested in finding a locally optimal design for a given set of parameters (listed above the model). In all cases, the initial grid is equidistributed having 101 points and $\epsilon = \epsilon_1 = 10^{-5}$, cf. Sect. 3.2. We also assess the effects of having different initial grid sets on the performance of our algorithm for finding optimal designs for Models 1 and 5. In Sect. 4.1, we test how well the extended algorithm generates $E-$optimal designs when $m_\lambda > 1$, and in Sect. 4.2 we report optimal designs found when there are two or more variables in the model, i.e., $n_x \geq 2$.

Tables 2, 3 and 4 present A-, D- and E-optimal designs for all the models in Table 1, with values in the first line representing the support points, $x_i,\ i \in [k]$ and values in the second line representing the corresponding weights, $w_i,\ i \in [k]$. The results are in good agreement with those found by other authors, when available, see Atkinson (2007), Pronzato and Zhigljavsky (2014) and Yu (2010a). The computation time required for solving all the problems is relatively short compared with the other algorithms, and in all cases, the proposed algorithms converge in 2 or 3 iterations, and as demonstrated in Sect. 3 finds globally optimal designs. The main difference in computation time is due to the need of additional iterations to reach the convergence criterion (17).

Table 1 Battery of one factor statistical models

Model	Regression function	Design space (**X**)
1	$\beta_0 + \beta_1\, x + \beta_2\, x^2$	$[-1, 1]$
2	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3$	$[-1, 1]$
3	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4$	$[-1, 1]$
4	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4 + \beta_5\, x^5$	$[-1, 1]$
5	$\beta_0 + \beta_1\, x + \beta_2\, x^{-1} + \beta_3\ \exp(-x)$	$[0.5, 2.5]$
6†	$\frac{V\ x}{k+x} + F\ x$	$[0, 2]$
	$V = 20, k = 0.05, F = 10$	
7‡	$E_0 + \frac{(E_\infty - E_0)\ x^m}{k_d^m + x^m}$	$[10^{-5}, 2]$
	$E_\infty = 1.70, E_0 = 0.137, k_d^m = 1, m = -1.5$	
8♯	$\gamma\ \exp[-\alpha\ (x-\theta) - \exp(-\lambda\ (x-\theta))]$	$[10, 50]$
	$\gamma = 1.946, \theta = 6.06, \alpha = 0.174, \lambda = 0.288$	

† Modified Michaelis–Menten model (López-Fidalgo et al. 2008).
‡ Four-parameter Hill model (Hill 1910).
♯ Coale-McNeil model (Coale and McNeil 1972)

Table 2 *D*-optimal designs for Models in Table 1, and the initial grid has 101 uniformly spaced points

Model	Design	CPU (s)	Iterations
1	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	4.10	2
2	$\begin{pmatrix} -1.0000, & -0.4500, & 0.45000, & 1.0000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	5.64	3
3	$\begin{pmatrix} -1.0000, & -0.6501, & 0.0000, & 0.6501, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	4.77	2
4	$\begin{pmatrix} -1.0000, & -0.7688, & -0.2900, & 0.2900, & 0.7688, & 1.0000 \\ 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667 \end{pmatrix}$	5.73	2
5	$\begin{pmatrix} 0.5000, & 0.7773, & 1.5800, & 2.5000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	4.95	2
6	$\begin{pmatrix} 0.0000, & 0.2699, & 2.0000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	7.18	3
7	$\begin{pmatrix} 10^{-5}, & 0.3042, & 0.9703, & 2.0000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	10.38	3
8	$\begin{pmatrix} 10.0000, & 13.1801, & 24.3901, & 50.0000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	5.86	2

The speed of the algorithm depends on two factors. First, the SDP-generated design usually has efficiency close to 1 and so provides an accurate initial solution. Consequently, the computed directional derivative of the design criterion evaluated at the SDP-generated design is accurate. This directional derivative in turn provides good candidate points for the new grid, determined by solving the NLP problem (15). For instance, Fig. 1b displays the dispersion function of the *D*-optimal design for Model 3 in successive iterations, and Fig. 2b displays the corresponding plot for the *A*-optimal design for Model 5. In both examples, we observe that the design resulting from the first iteration is close to the final optimal design. Second, all the candidate points for maximizing the dispersion function are determined simultaneously, and consequently, all points of the new grid are updated in a single step. This is different from other algorithms where the location of each point of the grid is updated sequentially, one at a time. Figures 1a and 2a show the grid evolution for both problems. We observe that most of the initial candidate points are discarded in the first iteration and the optimal design obtained with SDP includes only a few nodes located in the vicinity of the maxima of the dispersion function. We next apply the NLP procedure to find the local maximizers and use the distance checking procedure to collapse them when they are close. Afterward, the points remaining form the new grid, used to construct local FIM's, subsequently provided to SDP solver to determine an updated optimal design.

Table 3 *A*-optimal designs for Models in Table 1, and the initial grid has 101 uniformly spaced points

Model	Design	CPU (s)	Iterations
1	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2500, & 0.5000, & 0.2500 \end{pmatrix}$	4.03	2
2	$\begin{pmatrix} -1.0000, & -0.4667, & 0.4667, & 1.0000 \\ 0.1510, & 0.3490, & 0.3490, & 0.1510 \end{pmatrix}$	4.45	2
3	$\begin{pmatrix} -1.0000, & -0.6800, & 0.0000, & 0.68000, & 1.0000 \\ 0.1015, & 0.2504, & 0.2883, & 0.2504, & 0.1015 \end{pmatrix}$	4.14	2
4	$\begin{pmatrix} -1.0000, & -0.7902, & -0.2927, & 0.2927, & 0.7902, & 1.0000 \\ 0.0806, & 0.1880, & 0.2313, & 0.2313, & 0.1880, & 0.0806 \end{pmatrix}$	8.14	3
5	$\begin{pmatrix} 0.5000, & 0.7571, & 1.6718, & 2.5000 \\ 0.1546, & 0.3351, & 0.3452, & 0.1650 \end{pmatrix}$	12.22	2
6	$\begin{pmatrix} 0.0000, & 0.2401, & 2.0000 \\ 0.0209, & 0.7058, & 0.2733 \end{pmatrix}$	4.89	2
7	$\begin{pmatrix} 10^{-5}, & 0.2765, & 1.0301, & 2.0000 \\ 0.1474, & 0.2753, & 0.3518, & 0.2256 \end{pmatrix}$	9.72	3
8	$\begin{pmatrix} 10.0000, & 13.0001, & 24.3491, & 50.0000 \\ 0.3307, & 0.4593, & 0.1736, & 0.0365 \end{pmatrix}$	6.64	2

Table 4 *E*-optimal designs for Models in Table 1, and the initial grid has 101 uniformly spaced points

Model	Design	CPU (s)	Iterations
1	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2000, & 0.6000, & 0.2000 \end{pmatrix}$	3.70	2
2	$\begin{pmatrix} -1.0000, & -0.5000, & 0.5000, & 1.0000 \\ 0.1267, & 0.3733, & 0.3733, & 0.1267 \end{pmatrix}$	3.98	2
3	$\begin{pmatrix} -1.0000, & -0.7072, & 0.0000, & 0.7072, & 1.0000 \\ 0.0930, & 0.2481, & 0.3178, & 0.2481, & 0.0930 \end{pmatrix}$	6.17	3
4	$\begin{pmatrix} -1.0000, & -0.8090, & -0.3091, & 0.3091, & 0.8090, & 1.0000 \\ 0.0736, & 0.1804, & 0.2460, & 0.2460, & 0.1804, & 0.0736 \end{pmatrix}$	5.47	3
5	$\begin{pmatrix} 0.5000, & 0.7552, & 1.6800, & 2.5000 \\ 0.1486, & 0.3294, & 0.3541, & 0.1677 \end{pmatrix}$	5.97	3
6	$\begin{pmatrix} 0.0000, & 0.2201, & 2.0000 \\ 0.0253, & 0.7423, & 0.2324 \end{pmatrix}$	5.14	2
7	$\begin{pmatrix} 10^{-5}, & 0.2730, & 1.0530, & 2.0000 \\ 0.1288, & 0.2757, & 0.3631, & 0.2325 \end{pmatrix}$	9.11	3
8	$\begin{pmatrix} 10.0000, & 12.9563, & 25.3471, & 50.0000 \\ 0.3356, & 0.4640, & 0.1689, & 0.0315 \end{pmatrix}$	25–50	3

The comparison of the designs presented in Tables 2, 3 and 4 with designs reported in the literature obtained with other algorithms, for the cases where those are available, reveals that on average the efficiency of ours is above 99.9%, and there is a few cases where we obtained values above 100%, specifically for *E*-optimal designs. As for the CPU time, our algorithm is less efficient, on average it requires 7.71 times more resources than others used for testing [e.g., Yang (2013)], with values ranging from 1.55 to 15.41 times. However, in case a compiler is used the performance would be improved. We notice that in all our designs the initial SDP problem requires on average 36% of the total CPU time, corroborating the criticality of this step.

Now we analyze the impact of the initial grid on the optimal design found by the Algorithm 1. We do so by considering the *D*-optimality criterion for Model 3, and the *A*-optimality criterion for Model 5, and varying the number of points in the initial grid. In all cases, as is throughout the whole paper, grid points are all equidistributed. Table 5 shows the generated designs using different grid sets are very close, suggesting that the initial grid may have only a marginal impact on the optimal design. Initial coarser grids require more iterations to reach the convergence and so longer CPU time even though the initial SDP problem has fewer variables to solve. We also note that when the SDP-generated design has more points, the NLP procedure requires more computational time. We also observe *A*-optimal designs take

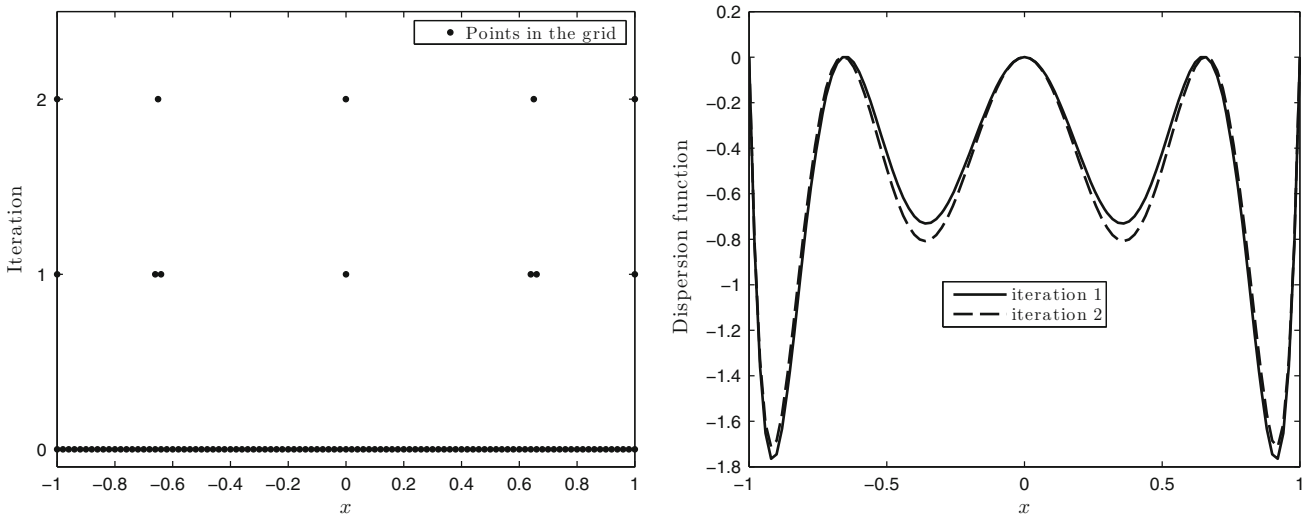

Fig. 1 The construction of the D-optimal design for Model 3 on $\mathbf{X} = [-1, 1]$ in Table 1 with $\Delta x = 0.02$: **a** grid evolution; **b** dispersion function evolution

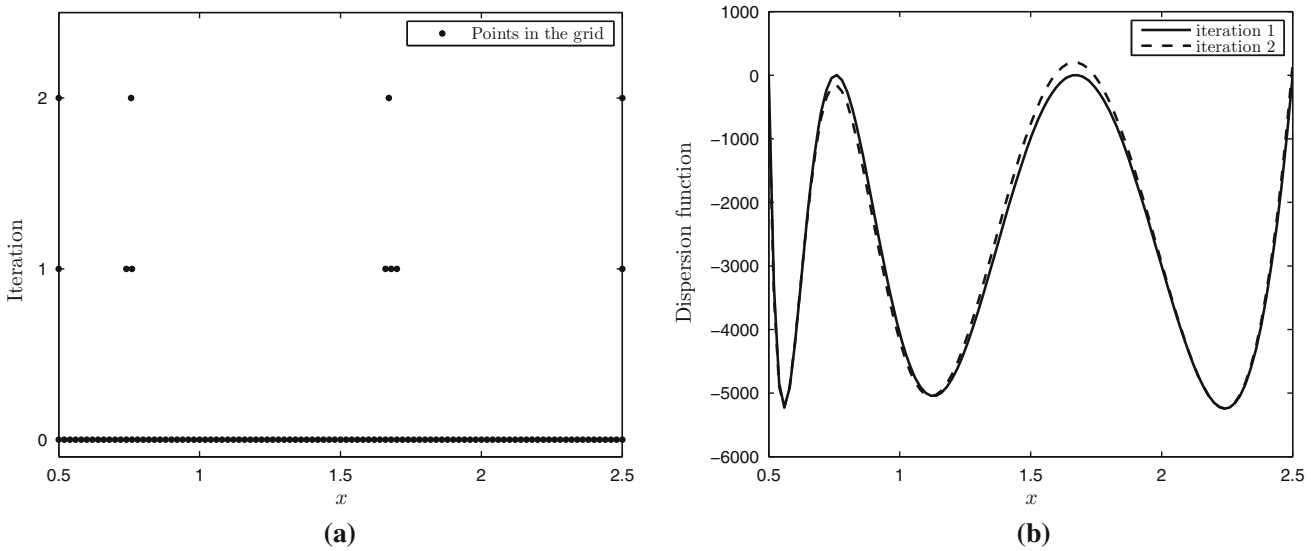

Fig. 2 The construction of the A-optimal design for Model 5 on $\mathbf{X} = [-1, 1]$ with $\Delta x = 0.02$: **a** grid evolution; **b** dispersion function evolution

longer time to find compared to other optimal designs even though an automatic scaling procedure has already been implemented.

4.1 E-optimal designs with $m_\lambda \geq 1$

E-optimal designs that have minimum eigenvalue with multiplicity greater than unity are harder to find and they commonly serve as benchmark tests for algorithms for continuous optimal designs of experiments. The verification of its global optimality is more difficult because a minmax problem needs to be solved so that the dispersion function is maximized in the design space for a feasible combination of the $\boldsymbol{\alpha}$'s. Our proposed algorithm is one of a few that can successfully handle the added complexity.

Table 6 lists the models we used to test the algorithm. They were selected in part because all of them yield FIM's where the minimum eigenvalue has multiplicity 2. The E-optimal designs for Models 1* and 2* were constructed by Melas (2006, Chap. 3) via functional analysis, and we compare them with those found by our proposed algorithm using their relative E- efficiencies defined by:

$$\text{Eff}_E = \frac{\lambda_{\min}(\mathscr{M}(\xi))}{\lambda_{\min}(\mathscr{M}(\xi^*))} \tag{20}$$

Table 5 A- and D-optimal designs found using different numbers of uniformly distributed points in the design space

Model	Criterion	Domain	Δx	Design	CPU (s)	Iterations
3	D-	$[-1, 1]$	0.01	$\begin{pmatrix} -1.0000, & -0.6550, & 0.0000, & 0.6550, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	5.25	2
			0.02	$\begin{pmatrix} -1.0000, & -0.6501, & 0.0000, & 0.6501, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	4.77	2
			0.04	$\begin{pmatrix} -1.0000, & -0.6594, & 0.0000, & 0.6594, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	6.31	3
			0.10	$\begin{pmatrix} -1.0000, & -0.6563, & 0.0000, & 0.6563, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	6.92	3
5	A-	$[0.5, 2.5]$	0.01	$\begin{pmatrix} 0.5000, & 0.7543, & 1.6698, & 2.5000 \\ 0.1561, & 0.3358, & 0.3439, & 0.1642 \end{pmatrix}$	10.66	3
			0.02	$\begin{pmatrix} 0.5000, & 0.7571, & 1.6718, & 2.5000 \\ 0.1546, & 0.3351, & 0.3452, & 0.1650 \end{pmatrix}$	12.22	2
			0.04	$\begin{pmatrix} 0.5000, & 0.7543, & 1.6720, & 2.5000 \\ 0.1560, & 0.3353, & 0.3440, & 0.1657 \end{pmatrix}$	11.66	3
			0.10	$\begin{pmatrix} 0.5000, & 0.7545, & 1.6672, & 2.5000 \\ 0.1562, & 0.3363, & 0.3439, & 0.1636 \end{pmatrix}$	21.06	5

Table 6 Models for which the minimum eigenvalue of the information matrix of the optimal design has multiplicity 2

Model	Regression function	Design space ($\mathbf{X}$)
1*	$\beta_0 + \beta_1\, x + \beta_2\, x^2$	$[-5, 5]$
2*	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3$	$[-5, 5]$
3*	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4$	$[-5, 5]$
4*	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4 + \beta_5\, x^5$	$[-5, 5]$

where $\mathscr{M}(\xi)$ is the FIM of the E-optimal design found from our algorithm and $\mathscr{M}(\xi^*)$ is the FIM of the optimal design found by Melas (2006, Chap. 3).

To analyze the details and the mechanics of the algorithm, let us consider the Model 2* in Table 6. The initial grid $\mathscr{G}^{(0)}$ is equidistributed and is formed by 201 points, corresponding to $\Delta x = 0.05$. The discrete domain-based optimal design $\xi^{(0)}$ has six points, two of them in the extremes of $\mathbf{X}$, two close to -1, and two in the vicinity of $+1$:

$$\xi^{(0)} = \begin{pmatrix} -5.0000, & -1.000, & -0.9500, & 0.9500, & 1.0000, & 5.0000 \\ 0.0184, & 0.2853, & 0.1963, & 0.1963, & 0.2853, & 0.0184 \end{pmatrix}$$

The design is symmetric and includes two pairs of neighbor support points. A simple analysis of $\xi^{(0)}$ reveals that the optimal design obtained considering a continuous space should have four support points, one in the interval $[-1.00, -0.95]$, the other in $[0.95, 1.00]$, plus the extremes. The minimum eigenvalue of the FIM of the design $\xi^{(0)}$ is 0.852267, and the corresponding eigenvalue for the design found by Melas (2006, Chap. 3) is 0.852281, which leads to $\text{Eff}_E = 0.99998$. It is noteworthy that the efficiency of the SDP initial design is high even considering that it includes two additional support points than the one obtained with functional analysis.

A direct calculation shows that $m_\lambda = 2$ for $\xi^{(0)}$ and the normalized eigenvectors associated with both $\lambda_{\min}$ are

$$\boldsymbol{e}_{\lambda_{\min},1} = \begin{pmatrix} -0.996814 \\ 0.000125 \\ 0.079750 \\ -0.000005 \end{pmatrix} \quad \text{and} \quad \boldsymbol{e}_{\lambda_{\min},2} = \begin{pmatrix} 0.000124 \\ 0.999138 \\ -0.000009 \\ -0.041509 \end{pmatrix}$$

and the dispersion functions constructed for each eigenvector $\boldsymbol{e}_{\lambda_{\min},i}$, $i \in \{1, 2\}$ are

$$\Psi_i(\boldsymbol{x}|\xi^{(0)}) = \text{tr}\left[\left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\mathsf{T}}\right)\, M(\delta_{\boldsymbol{x}})\right] - \lambda_{\min}, \quad i \in \{1, 2\},\ \boldsymbol{x} \in \mathbf{X}. \tag{21}$$

The dispersion function for the optimal convex matrix $\mathbf{E}$ is constructed by weighting the eigenvectors $\boldsymbol{e}_{\lambda_{\min},i}$ and is given by

$$\Psi_3(\boldsymbol{x}|\xi^{(0)}) = \text{tr}\left[\sum_{i=1}^{2} \alpha_i\, \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\mathsf{T}}\right)\, M(\delta_{\boldsymbol{x}})\right] - \lambda_{\min},\ \boldsymbol{x} \in \mathbf{X}, \tag{22}$$

where the vector of weights found by solving the mean absolute deviation problem (19) is $\boldsymbol{\alpha} = (0.85192, 0.14808)^{\mathsf{T}}$.

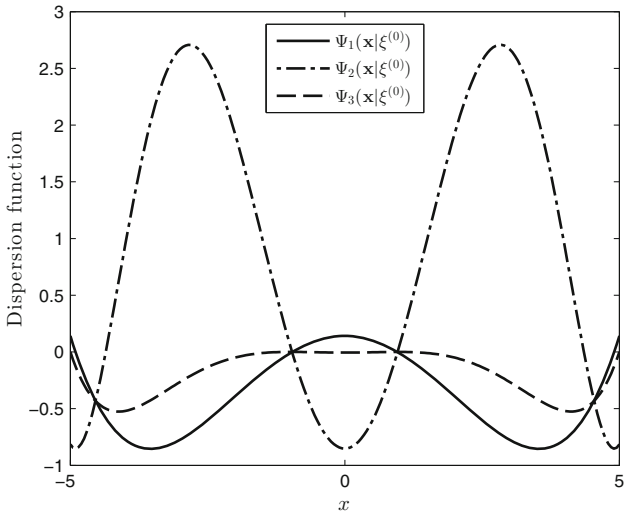

Fig. 3 The dispersion functions $\Psi_i(\boldsymbol{x}|\xi^{(0)})$, $i \in \{1, 2, 3\}$ of the design found from SDP in the first iteration for Model 2*

Figure 3 displays all three dispersion functions where $\Psi_3(\boldsymbol{x}|\xi^{(0)})$ and demonstrates the optimality of the design $\xi^{(0)}$. Next, the maxima of $\Psi_3(\boldsymbol{x}|\xi^{(0)})$ are determined solving the problem (15). The solution found is $\boldsymbol{x} = (-5.0000, -0.9783, -0.9783, 0.9783, 0.9783, 5.0000)^\intercal$ with the second and third points collapsing into one and the same is true for the fourth and fifth points. The grid of candidate points for the second iteration of the algorithm, $\mathcal{G}^{(1)}$, contains the four remaining points. This grid is then used to find a new design $\xi^{(1)}$ which is based on 4 support points, cf. Table 7, and the FIM has $\lambda_{\min} = 0.852154$ where $m_\lambda = 2$. This design satisfies the convergence condition (17), and the iteration procedure stops. Figure 4(a) illustrates the grid adaptation, and Fig. 4(b) presents the evolution of the dispersion function. We observe that the dispersion function constructed from $\xi^{(1)}$ is very similar to that of $\xi^{(0)}$ which strengthen the accuracy of the first design obtained with SDP although of having two points more than the latest.

The results in Table 7 show good agreement with those of Melas (2006, Chap. 3) and denote the ability of the algorithm to handle E-optimal designs with multiple minimum eigenvalues. The efficiency of the design found for Models 1* and 2* is 1.0000 and 0.9999, respectively, which corroborates the accuracy of the algorithm. We also observe that the algorithm requires more computation time than that used for similar models that lead to $m_\lambda = 1$ because of the additional linear program solved in each iteration.

4.2 Extension to higher-dimensional models

In this section, we consider linear models with $n_x \geq 2$. The extension to n_x-dimensional problems is straightforward, and Algorithm 1 does not need additional or special updates. The design space $\mathbf{X}$ is, in all cases but one, a Cartesian closed domain, and the candidate points included in $\mathcal{G}^{(0)}$ are the points of the n_x-dimensional grid obtained by equidistribution in each dimension. The space between the points is represented by $\Delta\boldsymbol{x}$ and can have different values for each dimension, so that for ith covariate x_i is $\Delta x_i = (x_i^{UP} - x_i^{LO})/(q_i - 1)$, and q_i is the number of points used in that particular dimension. The NLP problem to find the maxima of the dispersion function for a k-point design previously obtained with SDP has $k \times n_x$ variables and becomes computationally more challenging than that resulting from $n_x = 1$. However, nonlinear programs to solve in each iteration are small compared to those that `IPOPT` is capable of handling because the number of support points is low. All variables in the NLP program need to be bounded using constraints (15b, 15c and 15d). The gradient and the Jacobian matrix are still constructed by automatic differentiation, which now requires more computational time but is only performed once for each model.

The collapsing procedure deletes support points that are in the same ϵ-size ball belonging to $\mathbf{X}$. As in Sect. 4, the Euclidean distance is used to check whether two support points belong to the ϵ-size ball. For the ith iteration, we determine the matrix of distances between the support points using

$$d\left(\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_l^{(i)}\right) = \left\|\boldsymbol{x}_j^{(i)} - \boldsymbol{x}_l^{(i)}\right\|_2, \quad j,\ l \in [k^{(i)}] \tag{23}$$

and delete the l^{th} point when $d(\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_l^{(i)}) < \epsilon$.

Table 7 E-optimal designs for Models in Table 6, and the initial grid has 201 uniformly spaced points

Model	Design	CPU (s)	Iterations
1*	$\begin{pmatrix} -5.0000, & -0.0000, & 5.0000 \\ 0.0192, & 0.9616, & 0.0192 \end{pmatrix}$	5.03	2
2*	$\begin{pmatrix} -5.0000, & -0.9783, & 0.9783, & 5.0000 \\ 0.0184, & 0.4816, & 0.4816 & 0.0184 \end{pmatrix}$	6.53	2
3*	$\begin{pmatrix} -5.0000, & -2.6751, & 0.0000, & 2.6751, & 5.0000 \\ 0.0173, & 0.1131, & 0.7392, & 0.1130, & 0.0173 \end{pmatrix}$	6.23	2
4*	$\begin{pmatrix} -5.0000, & -3.6451, & -0.9257, & 0.9257, & 3.6451, & 5.0000 \\ 0.0194, & 0.0704, & 0.4102, & 0.4102, & 0.0704, & 0.0194 \end{pmatrix}$	6.98	2

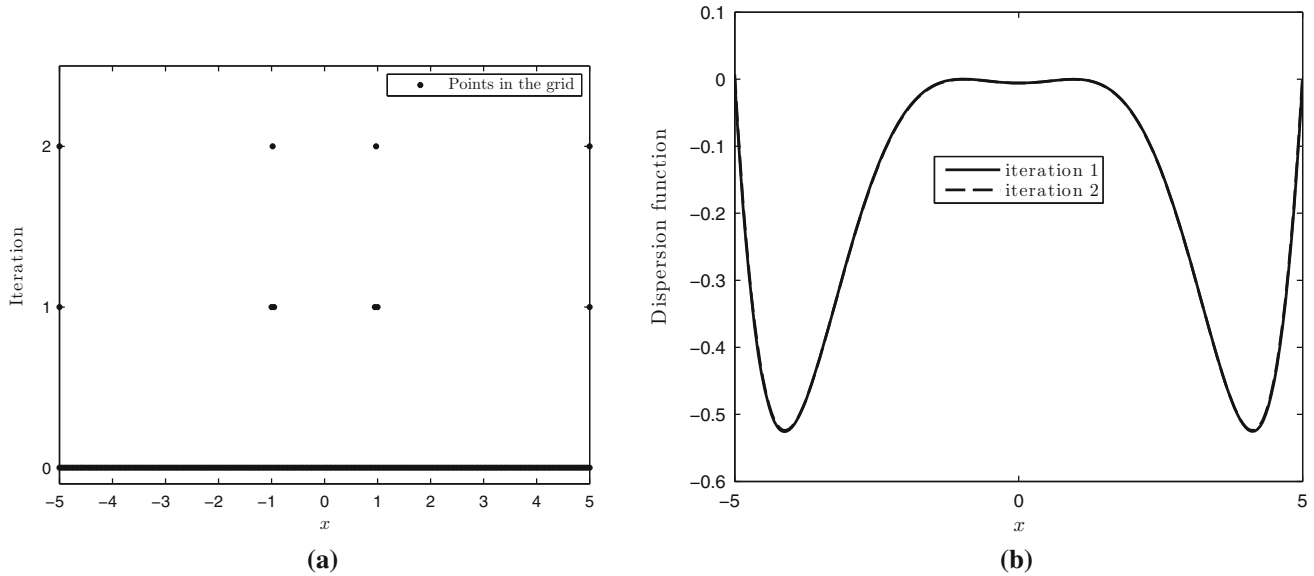

Fig. 4 The construction of the E-optimal design for Model 2* on $\mathbf{X} = [-5, 5]$ with $\Delta x = 0.05$: **a** grid evolution; **b** dispersion function evolution

Table 8 presents a battery of statistical linear and nonlinear models; Models 9–13 and 15–17 have two variables, and Models 14 and 18, 19 have three. To submit the algorithm to a broad range of problems, we address linear models (Models 9–14 and 17), and nonlinear models (Models 15, 16 and 18, 19), design spaces with the form of boxes (Models 9–18) and design spaces constrained by a simplex (Model 19). The linear models have regression functions including a wide range of terms, such as linear, quadratic, exponential, mixture and rational functions, commonly used to fit experimental data.

The design space for Models 9–13 is $\mathbf{X} = [-1, 1] \times [-1, 1]$, and 21 equidistributed points in each dimension are used to generate the initial grid which yields $\Delta \boldsymbol{x} = [0.05, 0.05]^\intercal$. Consequently, the initial grid is formed by 441 candidate points. The initial mesh can be coarser or thinner without major impact on the optimal design, as we observed in Sect. 4. Thinner initial grids may require a large amount of computational time to solve the initial SDP problem due to the size, and extremely coarse grids may require additional iterations to reach the convergence condition. We also use 21 equidistributed points in each dimension for Models 15–17, and note that for Model 15 $\Delta \boldsymbol{x} = [0.25,\ 0.05]^\intercal$.

The design space for Models 14 and 18 is $\mathbf{X} = [-1, 1] \times [-1, 1] \times [-1, 1]$, and we use 11 points to discretize each dimension. Consequently, the initial grid is formed by 1331 candidate points, and $\Delta \boldsymbol{x} = [0.1, 0.1, 0.1]^\intercal$. The design space of the Model 19 is discretized using 441 equidistributed points in each of the first two dimensions where $\Delta \boldsymbol{x} = [0.025, 0.025]^\intercal$. The values of x_3 for each point are obtained through the equality constraint.

Figure 5a, b displays the D- and A-optimal designs for Model 14. Both are symmetric having 27 points. Tables 9, 10 and 11 list the designs for all models, and we note the mild computational time required to solve each one. The results found are also in good agreement with those found with other algorithms, when those are available for comparison; see Atkinson (2007), Yu (2010a). All the designs obtained for the E-optimality criterion need the algorithm presented in Sect. 3.3 to handle FIM's where the multiplicity of the minimum eigenvalue is larger than 1; see column 3 of Table 11.

5 Summary

Our paper is the first to apply an algorithm that hybrids SDP with adaptive grid strategies to find optimal designs for linear models and locally optimal designs for nonlinear models. Our work is somewhat inspired by exchange methods and at the same time, take advantage of mathematical programming tools that guarantee optimality of the generated design in each step of convergence. The user first supplies an initial grid, and SDP is applied to find an optimal design on the grid. We then apply NLP to find points that maximize the dispersion function of the SDP-generated design, and they form the next grid set. The process is iterated until an ϵ-convergence condition is satisfied. We provided examples to show how our algorithm generates A- and D-optimal designs for polynomial and rational function models with one and multiple variables.

Maximin optimal design problems for general regression models are notoriously difficult to find, and we are not aware of algorithms that can systematically generate such designs. Using E-optimality as an illustrative example, we applied

Table 8 Battery of multiple factor statistical models

Model	Regression function	Design space ($\mathbf{X}$)
9	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\, x_1^2 + \beta_4\, x_2^2$	$[-1, 1] \times [-1, 1]$
10	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\, x_1^2 + \beta_4\, x_2^2 + \beta_5\, x_1\, x_2$	$[-1, 1] \times [-1, 1]$
11	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\ \exp(-x_1) + \beta_4\ \exp(-x_2)$	$[-1, 1] \times [-1, 1]$
12	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\ \exp(-x_1) + \beta_4\ \exp(-x_2) + \beta_5\, x_1\, x_2$	$[-1, 1] \times [-1, 1]$
13	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\ \exp(-x_1) + \beta_4\ \exp(-x_2) + \beta_5\ \exp(-x_1\, x_2)$	$[-1, 1] \times [-1, 1]$
14	$\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\, x_3 + \beta_4\, x_1^2 + \beta_5\, x_2^2 + \beta_6\, x_3^2 +$ $+ \beta_7\, x_1\, x_2 + \beta_8\, x_1\, x_3 + \beta_9\, x_2\, x_3$	$[-1, 1] \times [-1, 1] \times [-1, 1]$
15♣	$\frac{1}{1+\exp(\beta_0+\beta_1\, x_1+\beta_2\, x_2+\beta_3\, x_1\, x_2)}$ $\beta_0 = -2,\ \beta_1 = 0.5,\ \beta_2 = 0.5,\ \beta_3 = 0.1$	$[0, 5] \times [0, 1]$
16♭	$\frac{\theta_1\, \theta_3\, x_1}{1+\theta_1\, x_1+\theta_2\, x_2}$ $\theta_1 = 2.9,\ \theta_2 = 12.2,\ \theta_3 = 0.69$	$[0, 3] \times [0, 3]$
17	$\beta_0 + \beta_1\, x_1 + \beta_2\, x_2 + \beta_3\, \frac{x_1}{1+x_2} + \beta_4\, \frac{x_2}{1+x_1} + \beta_5\, \frac{x_1\, x_2}{1+x_1\, x_2}$	$[0, 1] \times [0, 1]$
18†	$\exp(\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\, x_3 + \beta_4\, x_1^2 + \beta_5\, x_2^2 + \beta_6\, x_3^2 +$ $+ \beta_7\, x_1\, x_2 + \beta_8\, x_1\, x_3 + \beta_9\, x_2\, x_3)$ $\beta_0 = 0.5, \beta_1 = -0.2,\ \beta_2 = 0.5,\ \beta_3 = -0.2,\ \beta_4 = -0.1,\ \beta_5 = 0.2,$ $\beta_6 = -0.1,\ \beta_7 = 0.2,\ \beta_8 = -0.1,\ \beta_9 = 0.2$	$[-1, 1] \times [-1, 1] \times [-1, 1]$
19‡	$\exp(\beta_0 + \beta_1\, x_1 + \beta_1\, x_2 + \beta_3\, x_3 + \beta_4\, x_1^2 + \beta_5\, x_2^2 + \beta_6\, x_3^2 +$ $+ \beta_7\, x_1\, x_2 + \beta_8\, x_1\, x_3 + \beta_9\, x_2\, x_3)$ $\beta_0 = 0.5, \beta_1 = -0.2,\ \beta_2 = 0.5,\ \beta_3 = -0.2,\ \beta_4 = 0.4,\ \beta_5 = -0.3,$ $\beta_6 = 0.7,\ \beta_7 = 0.7,\ \beta_8 = 1.7,\ \beta_9 = 1.0$	$[0, 0.5] \times [0, 0.5] \times$ $\times \{x_3 : x_1 + x_2 + x_3 = 1\}$

♣ Logistic model with two factors and all pairwise interactions.
♭ Kinetics of the catalytic dehydrogenation of *n*-hexil alcohol model (Box and Hunter 1965).
† Poisson model with 3 factors and all pairwise interactions.
‡ Poisson model with 3 factors and all pairwise interactions constrained to $\sum_{i=1}^{3} x_i = 1$

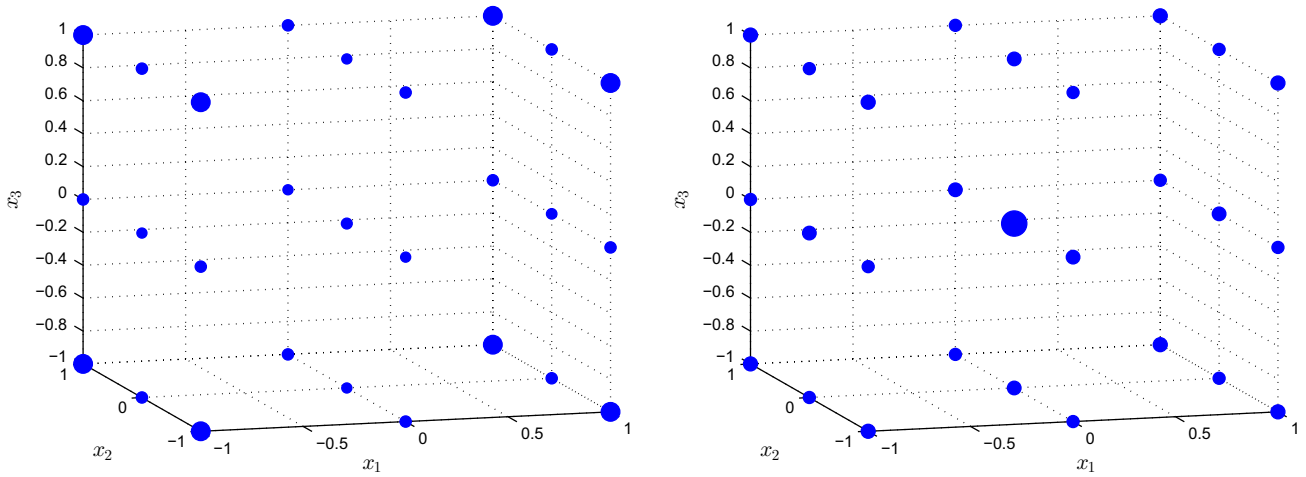

Fig. 5 **a** *D*-optimal; and **b** *A*-optimal designs for Model 25 in the domain $\mathbf{X} = [-1, 1] \times [-1, 1] \times [-1, 1]$ with $\Delta \boldsymbol{x} = [0.1, 0.1, 0.1]^\intercal$. The size of the markers is proportional to weight of the design at the particular point

our algorithm to find *E*-optimal designs for polynomial and rational function models with one or more variables. When the minimum eigenvalue of the FIM has multiplicity larger than 1, the design problem is more difficult because we have to work with subgradients. We showed our algorithm can also tackle such design problems systematically and does so by solving an additional LP problem that optimizes a convex combination of weights that validates the Equivalence The-

Table 9 *D*-optimal designs for Models in Table 8

Model	Design	CPU (s)	Iterations
9	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111 \end{pmatrix}$	5.33	2
10	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.1458, & 0.0802, & 0.1458, & 0.0802, & 0.0962, & 0.0802, & 0.1458, & 0.0802, & 0.1458 \end{pmatrix}$	6.08	2
11	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1650, & -0.1650, & -0.1650, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1650, & 1.0000, & -1.0000, & -0.1650, & 1.0000, & -1.0000, & -0.1650, & 1.0000 \\ 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111 \end{pmatrix}$	6.30	2
12	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1966, & -0.2027, & -0.1537, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1966, & 1.0000, & -1.0000, & 1.0000, & -0.1537, & -1.0000, & -0.2027, & 1.0000 \\ 0.1357, & 0.0834, & 0.1444, & 0.0834, & 0.0796, & 0.0956, & 0.1444, & 0.0796, & 0.1539 \end{pmatrix}$	7.17	2
13	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.2060, & -0.1484, & -0.1674, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.2060, & 1.0000, & -1.0000, & -0.1484, & 1.0000, & -1.0000, & -0.1674, & 1.0000 \\ 0.1324, & 0.0852, & 0.1480, & 0.0853, & 0.0977, & 0.0774, & 0.1480, & 0.0775, & 0.1486 \end{pmatrix}$	6.95	2
14	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0690, & 0.0249, & 0.0690, & 0.0249, & 0.0209, & 0.0249, & 0.0690, & 0.0249, & 0.0690 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0249, & 0.0209, & 0.0249, & 0.0209, & 0.0237, & 0.0209, & 0.0249, & 0.0209, & 0.0249 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0690, & 0.0249, & 0.0690, & 0.0249, & 0.0210, & 0.0249, & 0.0690, & 0.0249, & 0.0690 \end{array}\right)$	25.27	2
15	$\begin{pmatrix} 0.8000, & 1.4437, & 4.2362, & 5.0000 \\ 1.0000, & 0.0000, & 1.0000, & 0.0000 \\ 0.25000, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	12.00	3
16	$\begin{pmatrix} 0.3000, & 3.0000, & 3.0000 \\ 0.0000, & 0.0000, & 0.7944 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	7.87	2
17	$\begin{pmatrix} 0.0000, & 0.0000, & 0.6430, & 2.0000, & 2.0000, & 2.0000 \\ 0.0000, & 2.0000, & 2.0000, & 0.0000, & 0.6430, & 2.0000 \\ 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667 \end{pmatrix}$	7.03	2
18	$\left(\begin{array}{cccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & 0.0148, & 0.0139 \\ -1.0000, & -1.0000, & 0.0922, & 0.2195, & 1.0000, & 1.0000, & 1.0000, & 0.2195, & 1.0000 \\ -1.0000, & 1.0000, & -1.0000, & 0.0148, & -1.0000, & 0.0139, & 1.0000, & -1.0000, & -1.0000 \\ 0.0997, & 0.0971, & 0.0052, & 0.0641, & 0.0857, & 0.0341, & 0.0877, & 0.0641, & 0.0341 \\ \hline -0.0093, & -0.0332, & 1.0000, & 1.0000, & 1.0000, & 1.0000 & & & \\ 1.0000, & 1.0000, & -1.0000, & 1.0000, & 1.0000, & 1.0000 & & & \\ -0.0093, & 1.0000, & -1.0000, & -1.0000, & -0.0332, & 1.0000 & & & \\ 0.0715, & 0.0438, & 0.0971, & 0.0877 & 0.0437, & 0.0843 & & & \end{array}\right)$	39.81	3
19	$\begin{pmatrix} 0.0000, & 0.0000, & 0.0000, & 0.2485, & 0.2452, & 0.2420, & 0.5000, & 0.5000, & 0.5000 \\ 0.0000, & 0.2485, & 0.5000, & 0.0000, & 0.2452, & 0.5000, & 0.0000, & 0.2420, & 0.5000 \\ 1.0000, & 0.7515, & 0.5000, & 0.7515, & 0.5096, & 0.2580, & 0.5000, & 0.2580, & 0.0000 \\ 0.1440, & 0.0891, & 0.1453, & 0.0891, & 0.1014, & 0.0716, & 0.1453, & 0.0715, & 0.1426 \end{pmatrix}$	12.66	3

Table 10 *A*-optimal designs for Models in Table 8

Model	Design	CPU (s)	Iterations
9	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0704, & 0.1121, & 0.0704, & 0.1121, & 0.2698, & 0.1121, & 0.0704, & 0.1121, & 0.0704 \end{pmatrix}$	6.38	2
10	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0939, & 0.0978, & 0.0939, & 0.0978, & 0.2332, & 0.0978, & 0.0939, & 0.0978, & 0.0939 \end{pmatrix}$	5.52	2
11	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1566, & -0.1566, & -0.1566, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1566, & 1.0000, & -1.0000, & -0.1566, & 1.0000, & -1.0000, & -0.1566, & 1.0000 \\ 0.0943, & 0.0865, & 0.0484, & 0.0865, & 0.3328, & 0.1114, & 0.0484, & 0.1114, & 0.0803 \end{pmatrix}$	6.41	2
12	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1623, & -0.1506, & -0.1557, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1623, & 1.0000, & -1.0000, & -0.1506, & 1.0000, & -1.0000, & -0.1557, & 1.0000 \\ 0.0775, & 0.0848, & 0.0702, & 0.0848, & 0.3213, & 0.1096, & 0.0702, & 0.1096, & 0.0720 \end{pmatrix}$	7.06	2
13	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1613, & -0.1464, & -0.1422, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1613, & 1.0000, & -1.0000, & -0.1464, & 1.0000, & -1.0000, & -0.1422, & 1.0000 \\ 0.0886, & 0.0842, & 0.0571, & 0.0842, & 0.3196, & 0.1097, & 0.0571, & 0.1097, & 0.0899 \end{pmatrix}$	6.83	2
14	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0386, & 0.0291, & 0.0386, & 0.0291, & 0.0366, & 0.0291, & 0.0386, & 0.0291, & 0.0386 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0291, & 0.0366, & 0.0291, & 0.0366, & 0.1223, & 0.0366, & 0.0291, & 0.0366, & 0.0291 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0386, & 0.0291, & 0.0386, & 0.0291, & 0.0366, & 0.0291, & 0.0386, & 0.0291, & 0.0386 \end{array}\right)$	16.06	2
15	$\begin{pmatrix} 0.3537, & 0.7488, & 4.6460, & 5.0000 \\ 1.0000, & 0.0000, & 1.0000, & 0.0000 \\ 0.2957, & 0.5584, & 0.0660, & 0.0800 \end{pmatrix}$	12.25	3
16	$\begin{pmatrix} 0.2645, & 3.0000, & 3.0000 \\ 0.0000, & 0.0000, & 0.8022 \\ 0.4521, & 0.1079, & 0.4400 \end{pmatrix}$	9.02	3
17	$\begin{pmatrix} 0.0000, & 0.0000, & 0.6575, & 2.0000, & 2.0000, & 2.0000 \\ 0.0000, & 2.0000, & 2.0000, & 0.0000, & 0.6575, & 2.0000 \\ 0.0873, & 0.1178, & 0.2437, & 0.1178, & 0.2437, & 0.1896 \end{pmatrix}$	8.64	3
18	$\left(\begin{array}{ccccccccc} -0.0635, & -1.0000, & -1.0000, & -1.0000, & 0.3134, & 0.0163, & 0.2459, & 1.0000, & 0.8663 \\ -1.0000, & -1.0000, & 0.2940, & 0.2940, & 0.2940, & 1.0000, & -1.0000, & 0.2670, & 1.0000 \\ -1.0000, & 0.2589, & -1.0000, & 0.5923, & 1.0000, & 0.2468, & -1.0000, & -0.0635, & 1.0000 \\ 0.2798, & 0.0583, & 0.0410, & 0.0570, & 0.0805, & 0.0279, & 0.2831, & 0.0891, & 0.0370 \\ \hline 1.0000 \\ 0.2940 \\ -1.0000 \\ 0.0464 \end{array}\right)$	72.86	7
19	$\begin{pmatrix} 0.0000, & 0.0000, & 0.0000, & 0.2511, & 0.2445, & 0.2378, & 0.5000, & 0.5000, & 0.5000 \\ 0.0000, & 0.2511, & 0.5000, & 0.0000, & 0.2445, & 0.5000, & 0.0000, & 0.2378, & 0.5000 \\ 1.0000, & 0.7489, & 0.5000, & 0.7489, & 0.5110, & 0.2622, & 0.5000, & 0.2622, & 0.0000 \\ 0.1224, & 0.1233, & 0.0876, & 0.1233, & 0.1734, & 0.0925, & 0.0876, & 0.0925, & 0.0975 \end{pmatrix}$	10.42	3

Table 11 E-optimal designs for Models in Table 8

Model	Design	m_λ	CPU (s)	Iterations
9	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0500, & 0.1000, & 0.0500, & 0.1000, & 0.4000, & 0.1000, & 0.0500, & 0.1000, & 0.0500 \end{pmatrix}$	2	6.05	2
10	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0500, & 0.1000, & 0.0500, & 0.1000, & 0.4000, & 0.1000, & 0.0500, & 0.1000, & 0.0500 \end{pmatrix}$	3	6.67	2
11	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1614, & -0.1614, & -0.1614, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1614, & 1.0000 \\ 0.0954, & 0.0662, & 0.0693, & 0.0662, & 0.3943, & 0.08333, & 0.0693, & 0.0833, & 0.0724 \end{pmatrix}$	2	7.70	3
12	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1615, & -0.1614, & -0.1615, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1615, & 1.0000, & -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1615, & 1.0000 \\ 0.0908, & 0.0661, & 0.0740, & 0.0662, & 0.3941, & 0.0834, & 0.0740, & 0.0834, & 0.0679 \end{pmatrix}$	2	7.98	3
13	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1613, & -0.1617, & -0.1617, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1613, & 1.0000, & -1.0000, & -0.1617, & 1.0000, & -1.0000, & -0.1613, & 1.0000 \\ 0.1067, & 0.0662, & 0.0579, & 0.0662, & 0.3945, & 0.0834, & 0.0579, & 0.0834, & 0.0838 \end{pmatrix}$	2	8.22	3
14	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0135, & 0.0232, & 0.0135, & 0.0232, & 0.0537, & 0.0232, & 0.0135, & 0.0232, & 0.0135 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0232, & 0.0537, & 0.0232, & 0.0537, & 0.2924, & 0.0537, & 0.0232, & 0.0537, & 0.0232 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0135, & 0.0232, & 0.0135, & 0.0232, & 0.0537, & 0.0232, & 0.0135, & 0.0232, & 0.0135 \end{array}\right)$	6	20.38	3
15	$\begin{pmatrix} 0.0000, & 0.5385, & 5.0000, & 5.0000 \\ 1.0000, & 0.0000, & 0.0000, & 1.0000 \\ 0.3038, & 0.6237, & 0.0534, & 0.0190 \end{pmatrix}$	1	11.79	3
16	$\begin{pmatrix} 0.3000, & 2.7499, & 3.0000 \\ 0.2749, & 0.8000, & 1.1074 \\ 0.2066, & 0.3319, & 0.4615 \end{pmatrix}$	2	39.48	3
17	$\begin{pmatrix} 0.0000, & 0.0000, & 0.5759, & 2.0000, & 2.0000, & 2.0000 \\ 0.0000, & 2.0000, & 2.0000, & 0.0000, & 0.5736, & 2.0000 \\ 0.0807, & 0.0953, & 0.2478, & 0.0957, & 0.2480, & 0.2336 \end{pmatrix}$	2	14.39	2
18	$\left(\begin{array}{cccccccc} -1.0000, & -0.1469, & -1.0000, & -0.1922, & 0.1575, & -0.3605, & -0.1469, & 0.2000, & 1.0000 \\ -1.0000, & 0.1740, & 0.9964, & 1.0000, & 0.3440, & 1.0000, & -1.0000, & 0.0950, & 0.5139 \\ -0.7138, & -1.0000, & 1.0000, & 0.3491, & -0.2644, & 0.1657, & 0.6609, & -1.0000, & 0.6609 \\ 0.0100, & 0.1874, & 0.0483, & 0.1254, & 0.0866, & 0.0895, & 0.0086, & 0.2310, & 0.0498 \\ \hline 0.1575 & & & & & & & & \\ 1.0000 & & & & & & & & \\ 0.1657 & & & & & & & & \\ 0.1634 & & & & & & & & \end{array}\right)$	1	97.34	7
19	$\begin{pmatrix} 0.0000, & 0.0000, & 0.0000, & 0.2459, & 0.2412, & 0.2362, & 0.5000, & 0.5000, & 0.5000 \\ 0.0000, & 0.2459, & 0.5000, & 0.0000, & 0.2412, & 0.5000, & 0.0000, & 0.2362, & 0.5000 \\ 1.0000, & 0.7541, & 0.5000, & 0.7541, & 0.5176, & 0.2638, & 0.5000, & 0.2638, & 0.0000 \\ 0.0981, & 0.1392, & 0.0464, & 0.1391, & 0.2475, & 0.1095, & 0.0464, & 0.1095, & 0.0642 \end{pmatrix}$	2	8.88	3

orem. Our results are in good agreement with those obtained with other algorithms when those are available. A main difference is our algorithm is computationally efficient and is guaranteed to find the optimal design by construction.

Acknowledgements Dette's work was partially supported by the Collaborative Research Center "Statistical modeling of nonlinear dynamic processes" (SFB 823, Teilprojekt C2) of the German Research Foundation (DFG). Dette and Wong were also partially supported by a Grant from the National Institute of General Medical Sciences of the National Institutes of Health under Award Number R01GM107639. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Andersen, E., Jensen, B., Jensen, J., Sandvik, R., Worsøe, U.: Mosek version 6. Tech. rep., Technical Report TR-2009-3, MOSEK (2009)

Arthanari, T.S., Dodge, Y.: Mathematical Programming in Statistics. A Wiley-Interscience publication. Wiley, Hoboken (1993)

Atkinson, A.C., Donev, A.N.: The construction of exact *D*-optimum experimental designs with application to blocking response surface designs. Biometrika **76**(3), 515–526 (1989)

Atkinson, A.C., Donev, A.N., Tobias, R.D.: Optimum Experimental Designs, with SAS. Oxford University Press, Oxford (2007)

Ben-Tal, A., Nemirovski, A.S.: Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications. Society for Industrial and Applied Mathematics, Philadelphia (2001)

Berger, M.J.: Adaptive mesh refinement for hyperbolic partial differential equations. Ph.D. Dissertation, Department of Computer Science, Stanford University, Stanford, CA, USA (1982)

Berger, M.P.F., Wong, W.K.: An Introduction to Optimal Designs for Social and Biomedical Research. Wiley, Chichester (2009)

Bischof, C.H., Bücker, H.M., Lang, B., Rasch, A., Vehreschild, A.: Combining source transformation and operator overloading techniques to compute derivatives for Matlab programs. In: Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002), pp. 65–72. IEEE Computer Society, Los Alamitos, CA, USA (2002)

Box, G.E.P., Hunter, W.G.: The experimental study of physical mechanisms. Technometrics **7**(1), 23–42 (1965)

Boyd, S., Vandenberghe, L.: Convex Optimization. University Press, Cambridge (2004)

Byrd, R.H., Hribar, M.E., Nocedal, J.: An interior point algorithm for large-scale nonlinear programming. SIAM J. Optim. **9**(4), 877–900 (1999)

Chaloner, K., Larntz, K.: Optimal Bayesian design applied to logistic regression experiments. J. Stat. Plan. Inference **59**, 191–208 (1989)

Chen, R.B., Chang, S.P., Wang, W., Tung, H.C., Wong, W.K.: Minimax optimal designs via particle swarm optimization methods. Stat. Comput. **25**(5), 975–988 (2015)

Coale, A., McNeil, D.: The distribution by age of the frequency of first marriage in a female cohort. J. Am. Stat. Assoc. **67**(340), 743–749 (1972)

Coleman, T.F., Li, Y.: On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds. Math. Program. **67**(2), 189–224 (1994)

Cook, R., Nachtsheim, C.: Comparison of algorithms for constructing *D*-optimal design. Technometrics **22**(3), 315–324 (1980)

Dette, H., Grigoriev, Y.: E-optimal designs for second-order response surface models. Ann. Stat. **42**(4), 1635–1656 (2014)

Dette, H., Pepelyshev, A., Zhigljavsky, A.A.: Improving updating rules in multiplicative algorithms for computing *D*-optimal designs. Comput. Stat. Data Anal. **53**(2), 312–320 (2008)

Dette, H., Studden, W.J.: Geometry of *E*-optimality. Ann. Stat. **21**(1), 416–433 (1993)

Drud, A.: CONOPT: a GRG code for large sparse dynamic nonlinear optimization problems. Math. Program. **31**, 153–191 (1985)

Drud, A.: CONOPT—a large-scale GRG code. ORSA J. Comput. **6**(2), 207–216 (1994)

Duarte, B.P., Wong, W.K.: A semi-infinite programming based algorithm for finding minimax optimal designs for nonlinear models. Stat. Comput. **24**(6), 1063–1080 (2014)

Duarte, B.P.M., Wong, W.K.: Finding Bayesian optimal designs for nonlinear models: a semidefinite programming-based approach. Int. Stat. Rev. **83**(2), 239–262 (2015)

Duarte, B.P., Wong, W.K., Atkinson, A.C.: A semi-infinite programming based algorithm for determining T-optimum designs for model discrimination. J. Multivar. Anal. **135**, 11–24 (2015)

Fedorov, V.V.: Theory of Optimal Experiments. Academic Press, Cambridge (1972)

Fedorov, V.V., Leonov, S.L.: Optimal Design for Nonlinear Response Models. Chapman and Hall/CRC Press, Boca Raton (2014)

Fellman, J.: An empirical study of a class of iterative searches for optimal designs. J. Stat. Plan. Inference **21**, 85–92 (1989)

Filová, L., Trnovská, M., Harman, R.: Computing maximin efficient experimental designs using the methods of semidefinite programming. Metrika **64**(1), 109–119 (2011)

Gaivoronski, A.: Linearization methods for optimization of functionals which depend on probability measures. In: Prékopa, A., Wets, R.J.B. (eds.) Stochastic Programming 84 Part II, Mathematical Programming Studies, vol. 28, pp. 157–181. Springer, Berlin Heidelberg (1986)

Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005)

Goos, P., Jones, B.: Optimal Design of Experiments: A Case Study Approach. Wiley, New York (2011)

Grant, M., Boyd, S., Ye, Y.: cvx Users guide for cvx version 1.22. CVX Research, Inc., 1104 Claire Ave., Austin, TX 78703-2502 (2012)

Harman, R., Jurík, T.: Computing c-optimal experimental designs using the simplex method of linear programming. Comput. Stat. Data Anal. **53**(2), 247–254 (2008)

Harman, R., Pronzato, L.: Improvements on removing non-optimal support points in *D*-optimum design algorithms. Stat. Probab. Lett. **77**, 90–94 (2007)

Heredia-Langner, A., Montgomery, D.C., Carlyle, W.M., Borror, C.M.: Model-robust optimal designs: a genetic algorithm approach. J. Qual. Technol. **36**, 263–279 (2004)

Hill, A.: The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves. J Physiol **40**(Suppl), 4–7 (1910)

Kiefer, J.C.: Optimum experimental designs. J. R. Stat. Soc. Ser. B **21**, 272–319 (1959)

Kiefer, J.: General equivalence theory for optimum design (approximate theory). Ann. Stat. **2**, 849–879 (1974)

Kiefer, J., Wolfowitz, J.: The equivalence of two extremum problem. Can. J. Math. **12**, 363–366 (1960)

López-Fidalgo, J., Tommasi, C., Trandafir, P.C.: Optimal designs for discriminating between some extensions of the Michaelis–Menten model. J. Stat. Plan. Inference **138**(12), 3797–3804 (2008)

Mandal, S., Torsney, B.: Construction of optimal designs using a clustering approach. J. Stat. Plan. Inference **136**, 1120–1134 (2006)

Mandal, A., Wong, W.K., Yu, Y.: Algorithmic searches for optimal designs. Handbook of Design and Analysis of Experiments, chap 21, pp. 755–786. CRC Press, New York (2015)

Masoudi, E., Holling, H., Wong, W.K.: Application of imperialist competitive algorithm to find minimax and standardized maximin optimal designs. Comput. Stat. Data Anal. (in press). doi:10.1016/j.csda.2016.06.014

Melas, V.: Functional Approach to Optimal Experimental Design, Lecture Notes in Statistics. Springer (2006)

Meyer, R.K., Nachtsheim, C.J.: The coordinate-exchange algorithm for constructing exact optimal experimental designs. Technometrics **37**, 60–69 (1995)

Mitchell, T.J.: An algorithm for the construction of *D*-optimal experimental designs. Technometrics **16**, 203–210 (1974)

Mitchell, T.J., Miller Jr., F.L.: Use of design repair to construct designs for special linear models. Technical report, Oak Ridge National Laboratory, 130–131 (1970)

Molchanov, I., Zuyev, S.: Steepest descent algorithm in a space of measures. Stat. Comput. **12**, 115–123 (2002)

Papp, D.: Optimal designs for rational function regression. J. Am. Stat. Assoc. **107**, 400–411 (2012)

Pázman, A.: Foundations of Optimum Experimental Design (Mathematics and its Applications). Springer, Netherlands (1986)

Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.: Adaptive remeshing for compressible flow computations. J. Comput. Phys. **72**(2), 449–466 (1987)

Pronzato, L.: Removing non-optimal support points in *D*-optimum design algorithms. Stat. Probab. Lett. **63**(3), 223–228 (2003)

Pronzato, L.: Optimal experimental design and some related control problems. Automatica **44**, 303–325 (2008)

Pronzato, L., Zhigljavsky, A.A.: Algorithmic construction of optimal designs on compact sets for concave and differentiable criteria. J. Stat. Plan. Inference **154**, 141–155 (2014)

Pukelsheim, F.: Optimal Design of Experiments. SIAM, Philadelphia (1993)

Pukelsheim, F., Torsney, B.: Optimal weights for experimental designs on linearly independent support points. Ann. Stat. **19**(3), 1614–1625 (1991)

Qi, H.: A semidefinite programming study of the Elfving theorem. J. Stat. Plan. Inference **141**, 3117–3130 (2011)

Ruszczyński, A.P.: Nonlinear optimization, No. vol. 13. In: Nonlinear Optimization. Princeton University Press, Princeton (2006)

Sagnol, G.: Computing optimal designs of multiresponse experiments reduces to second-order cone programming. J. Stat. Plan. Inference **141**(5), 1684–1708 (2011)

Sagnol, G.: On the semidefinite representation of real functions applied to symmetric matrices. Linear Algebra Appl. **439**(10), 2829–2843 (2013)

Sagnol, G., Harman, R.: Computing exact *D*-optimal designs by mixed integer second order cone programming. Ann. Stat. **43**(5), 2198–2224 (2015)

Silvey, S.D.: Optimal Design. Chapman and Hall, London (1980)

Sturm, J.: Using `SeDuMi` 1.02, a Matlab toolbox for optimization over-symmetric cones. Optim Methods Softw. **11**, 625–653 (1999)

Titterington, D.M.: Algorithms for computing *D*-optimal design on finite design spaces. In: Proceedings of the 1976 Conference on Information Science and Systems, 3, John Hopkins University, 213–216 (1976)

Torsney, B., Mandal, S.: Two classes of multiplicative algorithms for constructing optimizing distributions. Comput. Stat. Data Anal. **51**(3), 1591–1601 (2006)

Torsney, B., Martín-Martín, R.: Multiplicative algorithms for computing optimum designs. J. Stat. Plan. Inference **139**(12), 3947–3961 (2009)

Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **8**, 49–95 (1996)

Vandenberghe, L., Boyd, S.: Applications of semidefinite programming. Appl. Numer. Math. **29**, 283–299 (1999)

Wächter, A., Biegler, T.L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2005)

Welch, W.J.: Algorithmic complexity: three NP-hard problems in computational statistics. J. Stat. Comput. Simul. **15**(1), 17–25 (1982)

Whittle, P.: Some general points in the theory of optimal experimental design. J. R. Stat. Soc. Ser. B **35**, 123–130 (1973)

Woods, D.C.: Robust designs for binary data: applications of simulated annealing. J. Stat. Comput. Simul. **80**(1), 29–41 (2010)

Wu, C.F.: Some algorithmic aspects of the theory of optimal designs. Ann. Stat. **6**(6), 1286–1301 (1978)

Wu, C.F., Wynn, H.P.: The convergence of general step-length algorithms for regular optimum design criteria. Ann. Stat. **6**(6), 1273–1285 (1978)

Wynn, H.P.: The sequential generation of *D*-optimum experimental designs. Ann. Math. Stat. **41**(5), 1655–1664 (1970)

Wynn, H.P.: Results in the theory and construction of *D*-optimum experimental designs. J. R. Stat. Soc. Ser. B **34**, 133–147 (1972)

Yang, M., Biedermann, S., Tang, E.: On optimal designs for nonlinear models: a general and efficient algorithm. J. Am. Stat. Assoc. **108**(504), 1411–1420 (2013)

Yu, Y.: *D*−optimal designs via a cocktail algorithm. Stat. Comput. **21**(4), 475–481 (2010a)

Yu, Y.: Monotonic convergence of a general algorithm for computing optimal designs. Ann. Stat. **38**(3), 1593–1606 (2010b)

Yu, Y.: Strict monotonicity and convergence rate of Titterington's algorithm for computing *D*-optimal designs. Comput. Stat. Data. Anal. **54**, 1419–1425 (2010c)