Adaptive grid Semidefinite Programming for finding optimal designs

Belmiro P.M. Duarte
Instituto Politécnico de Coimbra,
Instituto Superior de Engenharia de Coimbra,
Dep. of Chemical and Biological Engineering,
Rua Pedro Nunes, Quinta da Nora,
3030-199 Coimbra, Portugal.
and
CIEPQPF, Dep. of Chemical Engineering,
University of Coimbra, Coimbra, Portugal.
and
Weng Kee Wong
Department of Biostatistics,
Fielding School of Public Health, UCLA,
10833 Le Conte Ave.,
Los Angeles, California 90095-1772, U.S.A.
and
Holger Dette
Ruhr-Universitaet Bochum,
Department of Mathematics,
Institute of Statistics,
44780 Bochum, Germany.

June 13, 2016

Abstract

We find optimal designs for linear models using a novel algorithm that iteratively combines a Semidefinite Programming (SDP) approach with adaptive grid (AG) techniques. 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 (NLP). 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 models, we show the proposed algorithm can efficiently find both old and new optimal designs.

Keywords: Adaptive grid, Continuous design, Model-based optimal design, Nonlinear programming, Semidefinite programming.

1 Motivation

We consider the problem of determining model-based optimal designs of experiments (M-bODE) for algebraic 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). M-bODE is particularly helpful for providing maximum information at minimum cost. For M-bODE problems discussed in this paper, we assume we are given design criterion, a known *compact* design space and a known parametric linear model, apart from unknown parameters in the model. Typically the goal is to find an efficient design to estimate the model parameters.

Kiefer (1959) proposed viewing a M-bODE problem as 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 that optimally meet the design criterion (Atkinson et al., 2007). He termed these continuous designs and showed 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; the main reason is that the optimization problems can be complex and frequently fall into the NP-hard class (Welch, 1982). 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 optimal continuous designs for linear models efficiently.

During the last few decades, algorithms have been developed and continually improved for generating different types of optimal designs for algebraic models. Some examples are those proposed by Fedorov (1972), Wynn (1972), Mitchell (1974), Galil and Kiefer (1980) and recently, multiplicative algorithms seem to be gaining in popularity (Torsney and Mandal, 2006; Dette et al., 2008). Some of these algorithms are reviewed, compared and discussed in Cook and Nachtsheim (1982) and Pronzato (2008), among others. 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 vary 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 M-bODE 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). In this paper, we focus on SDP, which is not new; details on 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 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.

It is a curiosity that adaptive grid (AG) approaches have never been combined with mathematical programming formulations to find optimal designs, an exception is (Pronzato and Zhigljavsky, 2014, Sec. 3.4). Our algorithm is more general and includes exchange algorithms as special cases. In our proposed methodology, we have two levels of optimization: (i) the SDP solver finds the optimal design for a given grid; (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, which requires the solution of a constrained nonlinear program. We present an algorithm that automate the process and test it for linear 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 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 section 2.1 we introduce SDP as a tool to find optimal designs, and in section 2.2, we briefly review the fundamentals of NLP.

Throughout we assume we have a linear model with a given differentiable mean function $f(\boldsymbol{x})$ with linearly 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 the cartesian product of the domains of the variables and has dimension n_x and each x_i in $\boldsymbol{x} = (x_1, x_2, ..., x_{n_x})$ has a known range of interest indicated 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

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

where the vector of unknown model parameters is $\boldsymbol{p} \in \mathbf{P}$, 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 j^{th} parameter. The symbol $\mathbb{E}[\bullet]$ is the expectation operator. Given a design criterion and a predetermined sample size, n, our goal is to select the n sets of values for the variables to observe the responses. Replications are allowed and 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 $\boldsymbol{x}_i$, $i = 1, \cdots, k$ 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 i^{th} support point by $\boldsymbol{x}_i^\top = (x_{i,1}, \ldots, x_{i,n_x})$ and represent the design ξ by k rows $(\boldsymbol{x}_i^\top, w_i)$, $i \in \{1, \cdots, k\}$ with $\sum_{i=1}^{k} w_i = 1$. In what is to follow, we let $\Xi \equiv \mathbf{X}^k \times [0,1]^k$ be the space of feasible k-point designs over $\mathbf{X}$ and let $[k] = \{1, \cdots, k\}$.

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), $\mathscr{L}(\xi, \boldsymbol{p})$, with respect to the parameters, given by

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

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 $[q] = \{1, \cdots, q\}$ 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, with $D-$optimality, the volume of the confidence region of $\boldsymbol{p}$ is proportional to $\det[\mathscr{M}^{-1/2}(\xi)]$, and consequently maximizing the determinant (or its 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 follow: (i) ξ_D is D-optimal if and only if

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

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

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

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

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

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

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

Here $\boldsymbol{e}_{\lambda_{\min},1}, \dots, \boldsymbol{e}_{\lambda_{\min},m_\lambda}$ are normalized linearly independent eigenvectors of $\mathscr{M}(\xi_E)$ corresponding to $\lambda_{\min}$ with geometric multiplicity m_λ and $\alpha_1, \dots, \alpha_{m_\lambda}$ 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)-(7) as *dispersion functions*. They are different for different optimality concave criteria and we denote it by $\Psi(\boldsymbol{x}|\xi)$.

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 $\varphi : \mathbb{R}^{m_1} \mapsto \mathbb{R}$ 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 exists some symmetric matrices $M_0, \cdots, M_{m_1}, \cdots, M_{m_1+m_2} \in \mathbb{S}^{n_p}$ 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:

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

In our design context, $\boldsymbol{c}$ is a vector of known constants that depends on the design problem, and matrices M_i, $i = \{0, \cdots, m_1\}$ 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 corresponding to find a design for 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) $\mathbf{1}^\intercal \, \boldsymbol{w} = 1$.

A list of SDr functions was compiled by Ben-Tal and Nemirovski (2001, Chap. 2-3), and used for deriving SDP formulations for the M-bODE problem, see Boyd and Vandenberghe (2004, Sec. 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 is also applicable to the case where $\delta \to 0$.

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 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 $\boldsymbol{x} \in \mathbf{X}$ are points that we want to choose from to maximize 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 section 3.2.

Nested and Gradient Projection methods are commonly used to solve NP problems and they include 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). For an overview of NLP algorithms, the reader is referred to Ruszczyński (2006).

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 section 3.1 we introduce the formulation to find SDP-based designs, and in §3.2, we discuss the adaptive grid algorithm. Because the $E-$optimality criterion is not differentiable, the grid adaptation procedure has to be modified, particularly when the problem has multiple minimum eigenvalues; subsection 3.3 describes the strategy used 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

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

where diag($\mathscr{C}$) is the diagonal matrix with diagonal entries $\mathscr{C}_{j,j}$ and the geometric mean of the $\mathscr{C}_{j,j}$ on the extreme right 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:

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

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

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

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

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

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

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

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

We denote the design problems (12-14) by $\mathscr{P}_1$ and employ an 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 section 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 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). Practically, the grid $\mathscr{G}^{(1)}$ has $k^{(0)}$ potential points, but may be fewer if some of the solutions are the same or very close, in which case, they are collapsed to a single point.

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

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

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

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

where (15a) is the objective function and (15b-15d) are bound constraints for each maximum of the dispersion function, where the bounds are the support points of the previous SDP design. The problem (15) is designated $\mathscr{P}_2$, and two aspects are noteworthy. First, the dispersion function is often nonconvex and finding a single maximum is a challenging task. However, a linear combination of maxima (all with the same weight) and non overlapping due to the constraints (15b) makes the problem convex and easily tractable with NLP solvers. Second, each maximum is independent on the others so that the problem has a diagonal jacobian matrix.

We measure the distance between successive candidate points in $\boldsymbol{x}^{(1)}$ by

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

When they are ε-close for a pre-defined ε, i.e. $d(x_{i+1}^{(1)}, x_i^{(1)}) < \varepsilon$, the points are collapsed and a single point is included in the new grid $\mathscr{G}^{(1)}$; otherwise both points are included.

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:

$$\left|\frac{z^{(j)} - z^{(j-1)}}{z^{(j)}}\right| \leq \varepsilon_1 \tag{17}$$

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^*. This follows because (i) at each iteration the solver guarantees that the SDP-generated design is optimal and (ii) by construction, $\Phi_\delta[\mathscr{M}(\xi^{(j+1)})] \geq \Phi_\delta[\mathscr{M}(\xi^{(j)})]$ which is ensured because the grids in successive iterations, $\mathscr{G}^{(j+1)}$ and $\mathscr{G}^{(j)}$, respectively, are constructed such that $\max_{\boldsymbol{x}} \Psi(\boldsymbol{x}|\xi^{(j+1)}) \geq \max_{\boldsymbol{x}} \Psi(\boldsymbol{x}|\xi^{(j)})$, and consequently $\lim_{j\to\infty} \mathscr{G}^{(j)} \mapsto \mathscr{S}^*$, where $\mathscr{S}^*$ contains the set of support points of ξ^*. Since NLP global solvers are required to find the optimum of $\Psi(\boldsymbol{x}|\xi)$, the last inequality holds in every iteration.

All computation 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.

3.3 Adaptive strategy for finding $E-$optimal designs

Section 3.2 applied 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 non-zero 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 dimension 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 §2, cf. (7). Let $\lambda_{\min}^{(0)}$ be the minimum

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

```
procedure OPTIMALDESIGN($x^L$, $x^U$, $q$, $\varepsilon$, 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 $\Delta x$   ▷ 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 $|(z^{(j)} - z^{(j-1)})/z^{(j)}| > \varepsilon$ 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
```

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):

$$\min_{\boldsymbol{t},\boldsymbol{\alpha}} \sum_{l=1}^{k^{(0)}} t_l \tag{18a}$$

$$\text{s.t. } \operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^\intercal\right) M(\delta_{\boldsymbol{s}_l})\right] - \lambda_{\min}^{(0)} \le t_l, \quad \boldsymbol{s}_l \in \mathscr{S}^{(0)} \tag{18b}$$

$$\operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^\intercal\right) M(\delta_{\boldsymbol{s}_l})\right] - \lambda_{\min}^{(0)} \ge -t_l, \quad \boldsymbol{s}_l \in \mathscr{S}^{(0)} \tag{18c}$$

$$\operatorname{tr}\left[\sum_{i=1}^{m_\lambda} \alpha_i \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^\intercal\right) M(\delta_{\boldsymbol{x}_j})\right] - \lambda_{\min}^{(0)} \le 0, \quad \boldsymbol{x}_j \in \mathscr{G}^{(0)} \setminus \mathscr{S}^{(0)} \tag{18d}$$

$$\sum_{i=1}^{m_\lambda} \alpha_i = 1, \tag{18e}$$

$$\text{and } t_l \ge 0. \tag{18f}$$

Here (18b) and (18c) 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 to i^{th} smallest eigenvalue of $\mathscr{M}(\xi_E^{(0)})$. Equation (18d) 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 (18) 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 (18) using a tolerance level of 10^{-5}.

The next step in the extended algorithm computes the matrix $\mathbf{E}$ from $\boldsymbol{\alpha}$ using Equation (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 section 3.2 for $m_\lambda = 1$. If Algorithm 1 requires a few iterations to converge, problem (18) 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 §3 to find $D-$, $A-$ and $E-$optimal designs for a battery of linear models in Table 1. For models 1-7, the design space is $\mathbf{X} = [-1, 1]$ and for models 8-10, the design space is $\mathbf{X} = [0.5, 2.5]$. In all cases the initial grid is equidistributed having 101 points, $\Delta x = 0.02$ and $\varepsilon = \varepsilon_1 = 10^{-5}$, cf. §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 4 and 10. In section 4.1, we test how well the extended algorithm generates $E-$optimal designs when $m_\lambda > 1$, and in section 4.2 we report optimal designs found when there are 2 or more variables in the model, i.e. $n_x \ge 2$.

Table 1: Battery of linear statistical models.

Model	Form	Design space ($\mathbf{X}$)
1	$\beta_0 + \beta_1\, x$	$[-1, 1]$
2	$\beta_0 + \beta_1\, x + \beta_2\, x^2$	$[-1, 1]$
3	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3$	$[-1, 1]$
4	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4$	$[-1, 1]$
5	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4 + \beta_5\, x^5$	$[-1, 1]$
6	$\beta_0 + \beta_1\ \exp(x) + \beta_2\ \exp(-x)$	$[-1, 1]$
7	$\beta_0 + \beta_1\, x + \beta_2\ \exp(x) + \beta_3\ \exp(-x)$	$[-1, 1]$
8	$\beta_0 + \beta_1\, x + \beta_2\ \log(x)$	$[0.5, 2.5]$
9	$\beta_0 + \beta_1\, x + \beta_2\, x^{-1}$	$[0.5, 2.5]$
10	$\beta_0 + \beta_1\, x + \beta_2\, x^{-1} + \beta_3\ \exp(-x)$	$[0.5, 2.5]$

Tables 2-4 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, see (Atkinson et al., 2007; Pronzato and Zhigljavsky, 2014; Yu, 2010). The computation time required for solving all the problems is short compared with the other algorithms, and in all cases, the proposed algorithms converge in 2 or 3 iterations. The main difference in computation time are due to the need of additional iterations to reach the convergence criterion (17).

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, Figure 1(b) displays the dispersion function of the $D-$optimal design for Model 4 in successive iterations, and Figure 2(b) displays the corresponding plot for the $A-$optimal design for Model 10. In both examples we observe that the design resulting from the first iteration is close to the optimal solution. 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. Figure 1(a) and Figure 2(a) 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 support points and use the distance checking procedure to collapse them when they are close. Afterwards, 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 2: $D-$optimal designs for Models in Table 1, and the initial grid has 101 uniformly spaced points.

Model	Design	CPU (s)	Iterations
1	$\begin{pmatrix} -1.0000, & 1.0000 \\ 0.5000, & 0.5000 \end{pmatrix}$	4.08	2
2	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	4.10	2
3	$\begin{pmatrix} -1.0000, & -0.4500, & 0.45000, & 1.0000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	5.64	3
4	$\begin{pmatrix} -1.0000, & -0.6501, & 0.0000, & 0.6501, & 1.0000 \\ 0.2000, & 0.2000, & 0.2000, & 0.2000, & 0.2000 \end{pmatrix}$	4.77	2
5	$\begin{pmatrix} -1.0000, & -0.7688, & -0.2900, & 0.2900, & 0.7688, & 1.0000 \\ 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667, & 0.1667 \end{pmatrix}$	5.73	2
6	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	4.23	2
7	$\begin{pmatrix} -1.0000, & -0.4534, & 0.4534, & 1.0000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	5.61	2
8	$\begin{pmatrix} 0.5000, & 1.2435, & 2.5000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	5.80	2
9	$\begin{pmatrix} 0.5000, & 1.175, & 2.5000 \\ 0.3333, & 0.3333, & 0.3333 \end{pmatrix}$	4.56	2
10	$\begin{pmatrix} 0.5000, & 0.7773, & 1.5800, & 2.5000 \\ 0.2500, & 0.2500, & 0.2500, & 0.2500 \end{pmatrix}$	4.95	2

Now we analyze the impact of the initial grid on the optimal design found by the Algorithm 1. We consider the $D-$optimality criterion for Model 4, and the $A-$optimality criterion for Model 10, and varied the number of points of the initial grid which in all cases is 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 longer time to find compared to other optimal designs even though an automatic scaling procedure has already been implemented.

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	$\begin{pmatrix} -1.0000, & 1.0000 \\ 0.5000, & 0.5000 \end{pmatrix}$	3.67	2
2	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2500, & 0.5000, & 0.2500 \end{pmatrix}$	4.03	2
3	$\begin{pmatrix} -1.0000, & -0.4667, & 0.4667, & 1.0000 \\ 0.1510, & 0.3490, & 0.3490, & 0.1510 \end{pmatrix}$	4.45	2
4	$\begin{pmatrix} -1.0000, & -0.6800, & 0.0000, & 0.68000, & 1.0000 \\ 0.1015, & 0.2504, & 0.2883, & 0.2504, & 0.1015 \end{pmatrix}$	4.14	2
5	$\begin{pmatrix} -1.0000, & -0.7902, & -0.2927, & 0.2927, & 0.7902, & 1.0000 \\ 0.0806, & 0.1880, & 0.2313, & 0.2313, & 0.1880, & 0.0806 \end{pmatrix}$	8.14	3
6	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2138, & 0.5725, & 0.2138 \end{pmatrix}$	4.06	2
7	$\begin{pmatrix} -1.0000, & -0.4970, & 0.4970, & 1.0000 \\ 0.1606, & 0.3394, & 0.3394, & 0.1606 \end{pmatrix}$	5.38	2
8	$\begin{pmatrix} 0.5000, & 1.2370, & 2.5000 \\ 0.2426, & 0.5228, & 0.2346 \end{pmatrix}$	4.81	2
9	$\begin{pmatrix} 0.5000, & 1.172, & 2.5000 \\ 0.2329, & 0.5470, & 0.2202 \end{pmatrix}$	4.42	2
10	$\begin{pmatrix} 0.5000, & 0.7571, & 1.6718, & 2.5000 \\ 0.1546, & 0.3351, & 0.3452, & 0.1650 \end{pmatrix}$	12.22	2

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

$E-$optimal designs with the property that its minimum eigenvalue has multiplicity greater than unity are harder, and they commonly serve as benchmark tests for M-bODE algorithms. The verification of the global optimality is particularly 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 a few that can successfully handle the added complexity.

Table 6 lists the models for testing the algorithm, all of them yielding FIM's where the minimum eigenvalue has multiplicity 2. The $E-$optimal designs for models 2 and 3 were constructed by Melas (2006, Chap. 3) via functional analysis, and we compare them with those found by our proposed algorithm using their $E-$optimal

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	$\begin{pmatrix} -1.0000, & 1.0000 \\ 0.5000, & 0.5000 \end{pmatrix}$	3.80	2
2	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2000, & 0.6000, & 0.2000 \end{pmatrix}$	3.70	2
3	$\begin{pmatrix} -1.0000, & -0.5000, & 0.5000, & 1.0000 \\ 0.1267, & 0.3733, & 0.3733, & 0.1267 \end{pmatrix}$	3.98	2
4	$\begin{pmatrix} -1.0000, & -0.7072, & 0.0000, & 0.7072, & 1.0000 \\ 0.0930, & 0.2481, & 0.3178, & 0.2481, & 0.0930 \end{pmatrix}$	6.17	3
5	$\begin{pmatrix} -1.0000, & -0.8090, & -0.3091, & 0.3091, & 0.8090, & 1.0000 \\ 0.0736, & 0.1804, & 0.2460, & 0.2460, & 0.1804, & 0.0736 \end{pmatrix}$	5.47	3
6	$\begin{pmatrix} -1.0000, & 0.0000, & 1.0000 \\ 0.2093, & 0.5815, & 0.2093 \end{pmatrix}$	3.90	2
7	$\begin{pmatrix} -1.0000, & -0.5062, & 0.5062, & 1.0000 \\ 0.1588, & 0.3411, & 0.3411, & 0.1588 \end{pmatrix}$	5.14	2
8	$\begin{pmatrix} 0.5000, & 1.2427, & 2.5000 \\ 0.2351, & 0.5290, & 0.2359 \end{pmatrix}$	4.11	2
9	$\begin{pmatrix} 0.5000, & 1.1180, & 2.5000 \\ 0.2292, & 0.5531, & 0.2178 \end{pmatrix}$	4.09	2
10	$\begin{pmatrix} 0.5000, & 0.7552, & 1.6800, & 2.5000 \\ 0.1486, & 0.3294, & 0.3541, & 0.1677 \end{pmatrix}$	5.97	3

efficiencies defined by:

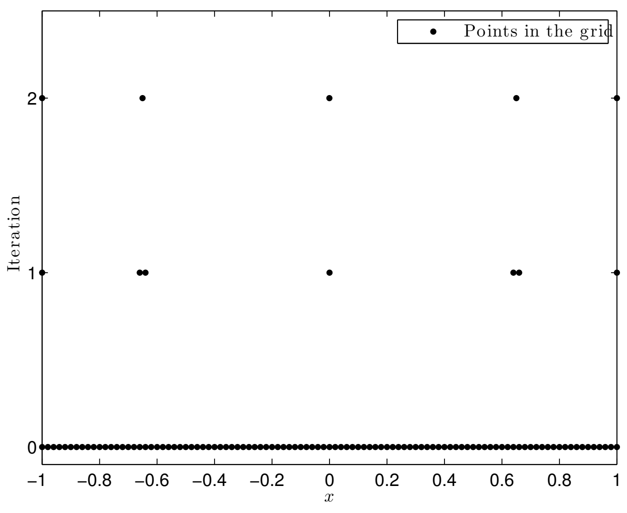
$$\text{Eff}_E = \frac{\lambda_{\min}(\mathscr{M}(\xi)}{\lambda_{\min}(\mathscr{M}(\xi^*)} \tag{19}$$

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

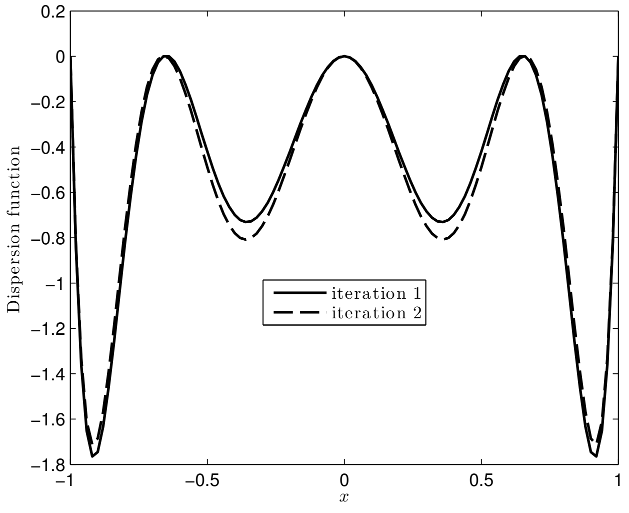
To analyze the details and the mechanics of the algorithm let us consider the Model 3 in Table 6. The initial grid $\mathscr{G}^{(0)}$ is equidistributed and is formed by 201 points. The discrete domain based optimal design $\xi^{(0)}$ has six points, two of them in the extremes of $\mathbf{X}$, two 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 $\xi^{(0)}$

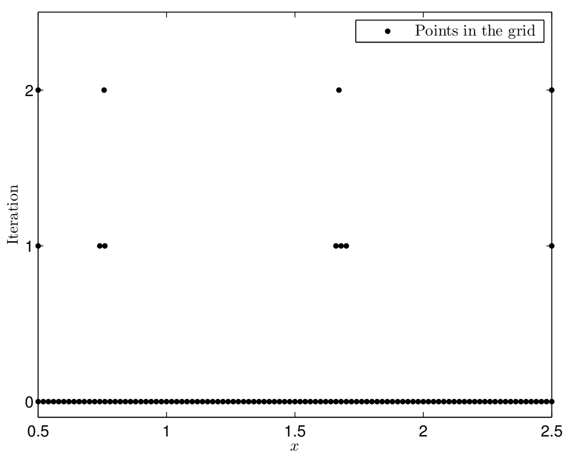


(a)

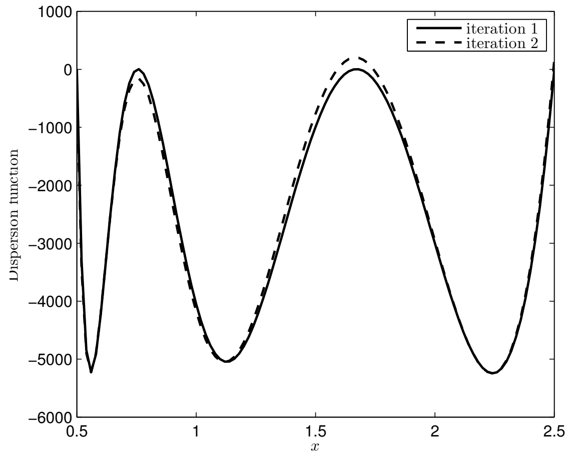


(b)

Figure 1: Construction of the $D-$optimal design for Model 4, $\mathbf{X} = [-1,1]$ and $\Delta x = 0.02$: (a) Grid evolution; (b) Dispersion function evolution.



(a)



(b)

Figure 2: Construction of the $A-$optimal design for Model 10, $\mathbf{X} = [-1,1]$ and $\Delta x = 0.02$: (a) Grid evolution; (b) Dispersion function evolution.

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.

The analysis of the FIM for $\xi^{(0)}$ also reveals that $m_\lambda = 2$, the normalized eigenvectors associated to both

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
4	$D-$	$[-1,1]$	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
10	$A-$	$[0.5,2.5]$	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	Form	Design space ($\mathbf{X}$)
2	$\beta_0 + \beta_1\, x + \beta_2\, x^2$	$[-5,5]$
3	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3$	$[-5,5]$
4	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4$	$[-5,5]$
5	$\beta_0 + \beta_1\, x + \beta_2\, x^2 + \beta_3\, x^3 + \beta_4\, x^4 + \beta_5\, x^5$	$[-5,5]$

$\lambda_{\min}$ are

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

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

$$\Psi_i(\boldsymbol{x}|\xi^{(0)}) = \text{tr}\left[\left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\intercal}\right)\, M(\delta_{\boldsymbol{x}})\right] - \lambda_{\min},\ i \in \{1,2\},\ \boldsymbol{x} \in \mathbf{X}. \tag{20}$$

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

$$\Psi_3(\boldsymbol{x}|\xi^{(0)}) = \text{tr}\left[\sum_{i=1}^{2} \alpha_i\, \left(\boldsymbol{e}_{\lambda_{\min},i}\, \boldsymbol{e}_{\lambda_{\min},i}^{\intercal}\right)\, M(\delta_{\boldsymbol{x}})\right] - \lambda_{\min},\ \boldsymbol{x} \in \mathbf{X} \tag{21}$$

The vector of weights found solving the mean absolute deviation problem (18) is $\boldsymbol{\alpha} = (0.85192, 0.14808)^{\intercal}$,

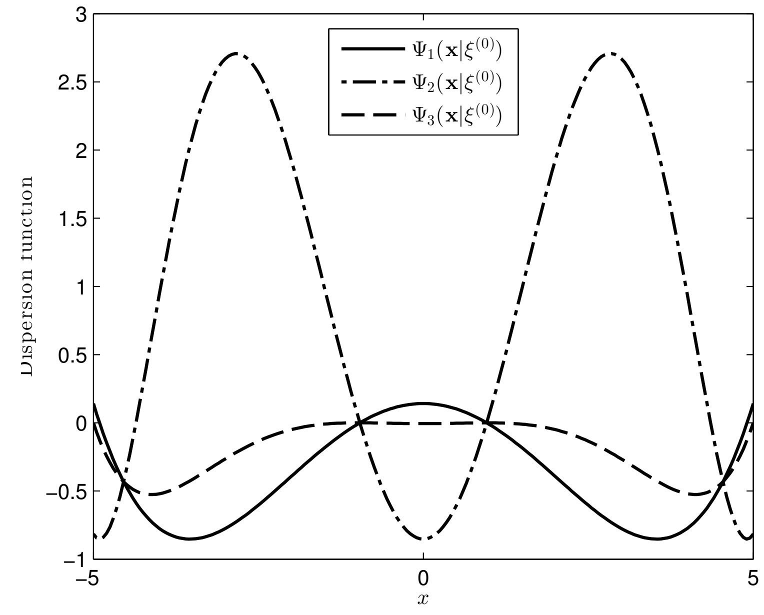


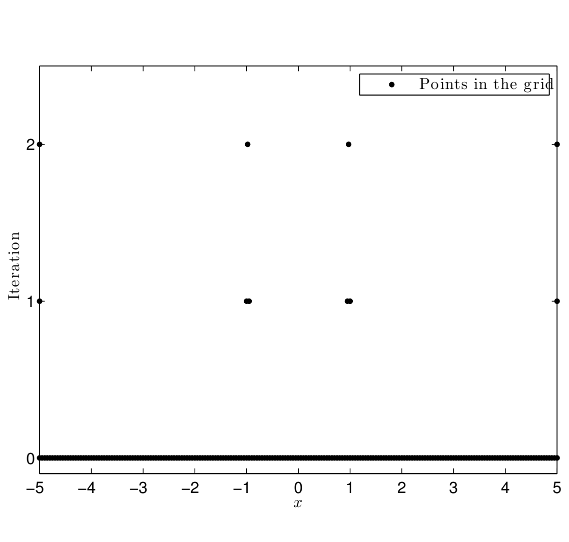
Figure 3: Dispersion functions $\Psi_i(\boldsymbol{x}|\xi^{(0)})$, $i \in \{1,2,3\}$ for the design obtained via SDP in the first iteration for Model 3.

and Figure 3 displays all three dispersion functions where $\Psi_3(\boldsymbol{x}|\xi^{(0)})$ 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 $\boldsymbol{x} = (-5.0000, -0.9783, -0.9783, 0.9783, 0.9783, 5.0000)^\intercal$, and subsequently second and third points collapse into one, the same occurring with fourth and fifth points. The grid of candidate points for the second iteration of the algorithm, $\mathscr{G}^{(1)}$, 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 $\lambda_{\text{min}} = 0.852154$ where $m_\lambda = 2$. This design satisfies the convergence condition (17), and the iteration procedure stops. The Figure 4(a) illustrates the grid adaptation, and Figure 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.

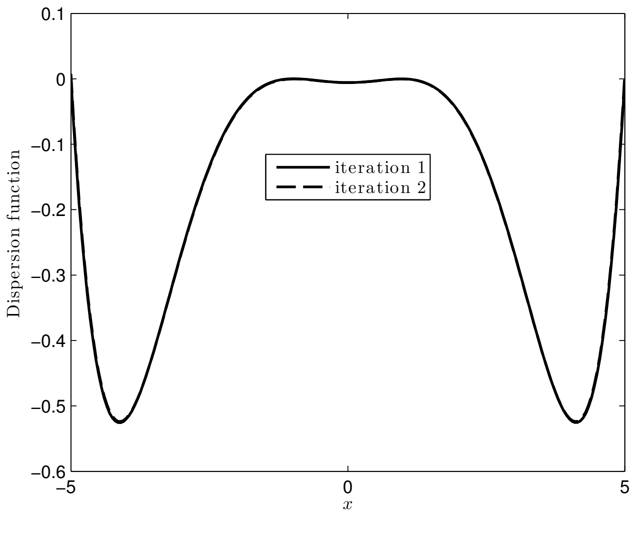
The results in Table 7 show good agreement with those of Melas (2006, Chap. 3) and denote the ability of the algorithm to handle $E-$optimal designs with multiple minimum eigenvalues. The efficiency of the design found for models 2 and 3 are 1.0000 and 0.9999, respectively, which corroborates the accuracy 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 dimensionality 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 a cartesian



(a)



(b)

Figure 4: Construction of the $E-$optimal design for Model 3, $\mathbf{X} = [-5,5]$ and $\Delta x = 0.05$: (a) Grid evolution; (b) Dispersion function evolution.

closed domain, and the candidate points included in $\mathscr{G}^{(0)}$ 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-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 section 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

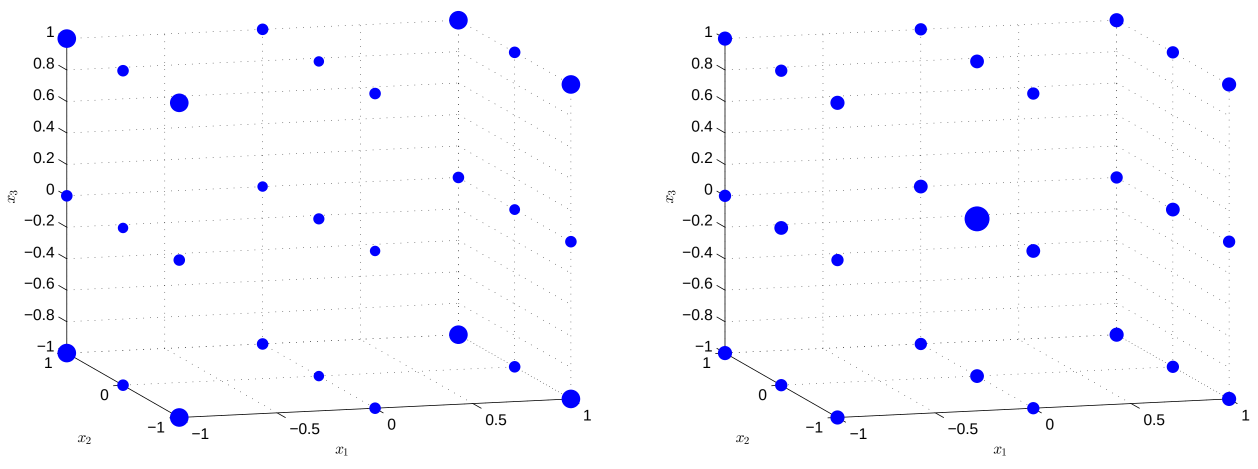
$$d(\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_l^{(i)}) = \left\| \boldsymbol{x}_j^{(i)} - \boldsymbol{x}_l^{(i)} \right\|_2, \quad j,\, l \in [k^{(i)}] \tag{22}$$

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

Table 8 presents a battery of statistical models, the first five (Models 20-24) have two variables, and the last (Model 25) has three. These linear models have regression functions with linear, quadratic, exponential and mixture terms, and they are commonly used to fit experimental data. The design space for Models 20-24 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 section 4.

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

Model	Design	CPU (s)	Iterations
2	$\begin{pmatrix} -5.0000, & -0.0000, & 5.0000 \\ 0.0192, & 0.9616, & 0.0192 \end{pmatrix}$	5.03	2
3	$\begin{pmatrix} -5.0000, & -0.9783, & 0.9783, & 5.0000 \\ 0.0184, & 0.4816, & 0.4816 & 0.0184 \end{pmatrix}$	6.53	2
4	$\begin{pmatrix} -5.0000, & -2.6751, & 0.0000, & 2.6751, & 5.0000 \\ 0.0173, & 0.1131, & 0.7392, & 0.1130, & 0.0173 \end{pmatrix}$	6.23	2
5	$\begin{pmatrix} -5.0000, & -3.6451, & -0.9257, & 0.9257, & 3.6451, & 5.0000 \\ 0.0194, & 0.0704, & 0.4102, & 0.4102, & 0.0704, & 0.0194 \end{pmatrix}$	6.98	2



(a) (b)

Figure 5: 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 weights of the design). (a) $D-$optimal design; (b) $A-$optimal design.

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.

The design space for Model 25 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 $\Delta\boldsymbol{x} = [0.1, 0.1, 0.1]^\intercal$. Figures 5(a) and 5(b) display the $D-$ and $A-$optimal designs for Model 25. Both are symmetric having 27 points. Tables 9-11 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, see (Atkinson et al., 2007; Yu, 2010). All the designs obtained for the $E-$optimality criterion need the algorithm presented in §3.3 to handle FIM's where the multiplicity of the minimum eigenvalue is larger than 1, see column 3 of Table 11.

Table 8: Model's code.

Model	Form	Design space ($\mathbf{X}$)
20	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,x_1^2+\beta_4\,x_2^2$	$[-1,1]\times[-1,1]$
21	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,x_1^2+\beta_4\,x_2^2+\beta_5\,x_1\,x_2$	$[-1,1]\times[-1,1]$
22	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,\exp(-x_1)+\beta_4\,\exp(-x_2)$	$[-1,1]\times[-1,1]$
23	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,\exp(-x_1)+\beta_4\,\exp(-x_2)+\beta_5\,x_1\,x_2$	$[-1,1]\times[-1,1]$
24	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,\exp(-x_1)+\beta_4\,\exp(-x_2)+\beta_5\,\exp(-x_1\,x_2)$	$[-1,1]\times[-1,1]$
25	$\beta_0+\beta_1\,x_1+\beta_1\,x_2+\beta_3\,x_3+\beta_4\,x_1^2+\beta_5\,x_2^2+\beta_6\,x_3^2+\beta_7\,x_1\,x_2+\beta_8\,x_1\,x_3+\beta_9\,x_2\,x_3$	$[-1,1]\times[-1,1]\times[-1,1]$

5 Summary

Our paper is the first to apply an algorithm that hybrids SDP with adaptive grid strategies to find globally optimal designs for linear 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. 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 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 our algorithm to find $E-$optimal designs for polynomial 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 Theorem. In all cases our results are in good agreement with those obtained with other algorithms. A main difference is our algorithm is computationally efficient and is guaranteed to find the optimal design by construction. We are currently extending the method to find optimal designs for nonlinear models.

Acknowledgments

This work has been supported in part 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 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.

Table 9: $D-$optimal designs for Models in Table 8.

Model	Design	CPU (s)	Iterations
20	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111 \end{pmatrix}$	5.33	2
21	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.1458, & 0.0802, & 0.1458, & 0.0802, & 0.0962, & 0.0802, & 0.1458, & 0.0802, & 0.1458 \end{pmatrix}$	6.08	2
22	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1650, & -0.1650, & -0.1650, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1650, & 1.0000, & -1.0000, & -0.1650, & 1.0000, & -1.0000, & -0.1650, & 1.0000 \\ 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111, & 0.1111 \end{pmatrix}$	6.30	2
23	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1966, & -0.2027, & -0.1537, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1966, & 1.0000, & -1.0000, & 1.0000, & -0.1537, & -1.0000, & -0.2027, & 1.0000 \\ 0.1357, & 0.0834, & 0.1444, & 0.0834, & 0.0796, & 0.0956, & 0.1444, & 0.0796, & 0.1539 \end{pmatrix}$	7.17	2
24	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.2060, & -0.1484, & -0.1674, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.2060, & 1.0000, & -1.0000, & -0.1484, & 1.0000, & -1.0000, & -0.1674, & 1.0000 \\ 0.1324, & 0.0852, & 0.1480, & 0.0853, & 0.0977, & 0.0774, & 0.1480, & 0.0775, & 0.1486 \end{pmatrix}$	6.95	2
25	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0690, & 0.0249, & 0.0690, & 0.0249, & 0.0209, & 0.0249, & 0.0690, & 0.0249, & 0.0690 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0249, & 0.0209, & 0.0249, & 0.0209, & 0.0237, & 0.0209, & 0.0249, & 0.0209, & 0.0249 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0690, & 0.0249, & 0.0690, & 0.0249, & 0.0210, & 0.0249, & 0.0690, & 0.0249, & 0.0690 \end{array}\right)$	25.27	2

References

Andersen, E., B. Jensen, J. Jensen, R. Sandvik, and U. Worsøe (2009). Mosek version 6. Technical report, Technical Report TR–2009–3, MOSEK.

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

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

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

Berger, M. J. (1982, August). *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*. Ph.D. dissertation, Department of Computer Science, Stanford University, Stanford, CA, USA.

Table 10: $A-$optimal designs for Models in Table 8.

Model	Design	CPU (s)	Iterations
20	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0704, & 0.1121, & 0.0704, & 0.1121, & 0.2698, & 0.1121, & 0.0704, & 0.1121, & 0.0704 \end{pmatrix}$	6.38	2
21	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0939, & 0.0978, & 0.0939, & 0.0978, & 0.2332, & 0.0978, & 0.0939, & 0.0978, & 0.0939 \end{pmatrix}$	5.52	2
22	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1566, & -0.1566, & -0.1566, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1566, & 1.0000, & -1.0000, & -0.1566, & 1.0000, & -1.0000, & -0.1566, & 1.0000 \\ 0.0943, & 0.0865, & 0.0484, & 0.0865, & 0.3328, & 0.1114, & 0.0484, & 0.1114, & 0.0803 \end{pmatrix}$	6.41	2
23	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1623, & -0.1506, & -0.1557, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1623, & 1.0000, & -1.0000, & -0.1506, & 1.0000, & -1.0000, & -0.1557, & 1.0000 \\ 0.0775, & 0.0848, & 0.0702, & 0.0848, & 0.3213, & 0.1096, & 0.0702, & 0.1096, & 0.0720 \end{pmatrix}$	7.06	2
24	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1613, & -0.1464, & -0.1422, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1613, & 1.0000, & -1.0000, & -0.1464, & 1.0000, & -1.0000, & -0.1422, & 1.0000 \\ 0.0886, & 0.0842, & 0.0571, & 0.0842, & 0.3196, & 0.1097, & 0.0571, & 0.1097, & 0.0899 \end{pmatrix}$	6.83	2
25	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0386, & 0.0291, & 0.0386, & 0.0291, & 0.0366, & 0.0291, & 0.0386, & 0.0291, & 0.0386 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0291, & 0.0366, & 0.0291, & 0.0366, & 0.1223, & 0.0366, & 0.0291, & 0.0366, & 0.0291 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0386, & 0.0291, & 0.0386, & 0.0291, & 0.0366, & 0.0291, & 0.0386, & 0.0291, & 0.0386 \end{array}\right)$	16.06	2

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

Bischof, C. H., H. M. Bücker, B. Lang, A. Rasch, and A. Vehreschild (2002). 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)*, Los Alamitos, CA, USA, pp. 65–72. IEEE Computer Society.

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

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

Chaloner, K. and K. Larntz (1989). Optimal Bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inference 59*, 191–208.

Table 11: $E-$optimal designs for Models in Table 8.

Model	Design	m_λ	CPU (s)	Iterations
20	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0500, & 0.1000, & 0.0500, & 0.1000, & 0.4000, & 0.1000, & 0.0500, & 0.1000, & 0.0500 \end{pmatrix}$	2	6.05	2
21	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0500, & 0.1000, & 0.0500, & 0.1000, & 0.4000, & 0.1000, & 0.0500, & 0.1000, & 0.0500 \end{pmatrix}$	3	6.67	2
22	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1614, & -0.1614, & -0.1614, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1614, & 1.0000 \\ 0.0954, & 0.0662, & 0.0693, & 0.0662, & 0.3943, & 0.08333, & 0.0693, & 0.0833, & 0.0724 \end{pmatrix}$	2	7.70	3
23	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1615, & -0.1614, & -0.1615, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1615, & 1.0000, & -1.0000, & -0.1614, & 1.0000, & -1.0000, & -0.1615, & 1.0000 \\ 0.0908, & 0.0661, & 0.0740, & 0.0662, & 0.3941, & 0.0834, & 0.0740, & 0.0834, & 0.0679 \end{pmatrix}$	2	7.98	3
24	$\begin{pmatrix} -1.0000, & -1.0000, & -1.0000, & -0.1613, & -0.1617, & -0.1617, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -0.1613, & 1.0000, & -1.0000, & -0.1617, & 1.0000, & -1.0000, & -0.1613, & 1.0000 \\ 0.1067, & 0.0662, & 0.0579, & 0.0662, & 0.3945, & 0.0834, & 0.0579, & 0.0834, & 0.0838 \end{pmatrix}$	2	8.22	3
25	$\left(\begin{array}{ccccccccc} -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000, & -1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0135, & 0.0232, & 0.0135, & 0.0232, & 0.0537, & 0.0232, & 0.0135, & 0.0232, & 0.0135 \\ \hline 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000, & 0.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0232, & 0.0537, & 0.0232, & 0.0537, & 0.2924, & 0.0537, & 0.0232, & 0.0537, & 0.0232 \\ \hline 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & -1.0000, & -1.0000, & 0.0000, & 0.0000, & 0.0000, & 1.0000, & 1.0000, & 1.0000 \\ -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000, & -1.0000, & 0.0000, & 1.0000 \\ 0.0135, & 0.0232, & 0.0135, & 0.0232, & 0.0537, & 0.0232, & 0.0135, & 0.0232, & 0.0135 \end{array}\right)$	6	20.38	3

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

Cook, R. D. and C. J. Nachtsheim (1982). Model robust, linear-optimal designs. *Technometrics 24*, 49–54.

Dette, H. and Y. Grigoriev (2014). E-optimal designs for second-order response surface models. *Annals of Statistics 42*(4), 1635–1656.

Dette, H., A. Pepelyshev, and A. A. Zhigljavsky (2008). Improving updating rules in multiplicative algorithms for computing D-optimal designs. *Computational Statistics & Data Analysis 53*(2), 312–320.

Dette, H. and W. J. Studden (1993, 03). Geometry of E-optimality. *Ann. Statist. 21*(1), 416–433.

Drud, A. (1985). `CONOPT`: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming 31*, 153–191.

Drud, A. (1994). `CONOPT` - A large–scale GRG code. *ORSA Journal on Computing 6*(2), 207–216.

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

Duarte, B. P., W. K. Wong, and A. C. Atkinson (2015). A semi-infinite programming based algorithm for determining T-optimum designs for model discrimination. *Journal of Multivariate Analysis 135*, 11 – 24.

Duarte, B. P. M. and W. K. Wong (2015). Finding Bayesian optimal designs for nonlinear models: A semidefinite programming-based approach. *International Statistical Review 83*(2), 239–262.

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

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

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

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

Galil, Z. and J. Kiefer (1980). Time- and space-saving computer methods, related to Mitchell's DETMAX for finding D-optimum designs. *Technometrics 22*, 301–313.

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

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

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

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

Kiefer, J. (1974). General equivalence theory for optimum design (approximate theory). *Annals of Statistics 2*, 849–879.

Kiefer, J. and J. Wolfowitz (1960). The equivalence of two extremum problem. *Canadian Journal of Mathematics 12*, 363–366.

Kiefer, J. C. (1959). Optimum experimental designs. *Journal of the Royal Statistical Society, Series B 21*, 272–319.

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

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

Molchanov, I. and S. Zuyev (2002). Steepest descent algorithm in a space of measures. *Statistics and Computing 12*, 115–123.

Papp, D. (2012). Optimal designs for rational function regression. *Journal of the American Statistical Association 107*, 400–411.

Peraire, J., M. Vahdati, K. Morgan, and O. Zienkiewicz (1987). Adaptive remeshing for compressible flow computations. *Journal of Computational Physics 72*(2), 449 – 466.

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

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

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

Qi, H. (2011). A semidefinite programming study of the Elfving theorem. *Journal of Statistical Planning and Inference 141*, 3117–3130.

Ruszczyński, A. P. (2006). *Nonlinear Optimization*. Number vol. 13 in Nonlinear optimization. Princeton University Press.

Sagnol, G. (2011). Computing optimal designs of multiresponse experiments reduces to second-order cone programming. *Journal of Statistical Planning and Inference 141*(5), 1684–1708.

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

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

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

Sturm, J. (1999). Using `SeDuMi` 1.02, a Matlab toolbox for optimization oversymmetric cones. *Optimization Methods and Software 11*, 625–653.

Torsney, B. and S. Mandal (2006). Two classes of multiplicative algorithms for constructing optimizing distributions. *Computational Statistics & Data Analysis 51*(3), 1591–1601.

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

Vandenberghe, L. and S. Boyd (1999). Applications of semidefinite programming. *Applied Numerical Mathematics 29*, 283–299.

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

Welch, W. J. (1982). Algorithmic complexity: Three NP-hard problems in computational statistics. *Journal of Statistical Computation and Simulation 15*(1), 17–25.

Whittle, P. (1973). Some general points in the theory of optimal experimental design. *Journal of the Royal Statistical Society, Ser. B 35*, 123–130.

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

Yu, Y. (2010). D-optimal designs via a cocktail algorithm. *Statistics and Computing 21*(4), 475–481.