Received: 13 September 2024 | Accepted: 17 September 2024

DOI: 10.1002/qre.3660

RESEARCH ARTICLE

WILEY

A non-approximate method for generating *G*-optimal RSM designs

Hyrum J. Hansen[1] | **Stephen J. Walsh**[1] | **Rong Pan**[2]

[1]Department of Mathematics and Statistics, Utah State University, Logan, Utah, USA

[2]School of Computing and Augmented Intelligence, Arizona State University, Tempe, Arizona, USA

Correspondence
Stephen J. Walsh, Department of Mathematics and Statistics, Utah State University, Utah, USA.
Email: steve.walsh@usu.edu

Abstract

We present a nonapproximate computational method for generating *G*-optimal designs in response surface methodology (RSM) settings using `Gloptipoly`, a global polynomial optimizer. Traditional approaches use a grid approximation for computing a candidate design's *G*-score. `Gloptipoly` can find the global optimum of high-order polynomials thus making it suitable for computing a design's *G*-score, that is, its maximum scaled prediction variance, which, for second-order models, is a quartic polynomial function of the experimental factors. We demonstrate the efficacy and performance of our method through comprehensive application to well-published examples, and illustrate, for the first time, its application to generating *G*-optimal designs supporting models of order greater than 2. This work represents the first non-approximate computational approach to solving the *G*-optimal design problem. This advancement opens new possibilities for finding *G*-optimal designs beyond second-order RSM models.

KEYWORDS

G-optimal design, `Gloptipoly`, polynomial optimization, response surface methodology

1 | INTRODUCTION

Generating optimal experimental designs that enable precise prediction in untested areas of the design space is a critical component of response surface methodology (RSM).[1] *G*-optimal designs, that is, those that minimize the model's maximum prediction variance, are thus attractive at the optimization stage of RSM. However, finding these designs remains a complicated technical challenge and continues to stimulate research. The most efficacious methods utilize a grid approximation to compute the *G*-score of a candidate design, leading to potential inaccuracies. We present a novel approach to generating *G*-optimal designs by utilizing `Gloptipoly`,[2,3] a sophisticated measure-theoretic global polynomial optimizer, to compute the *G*-score of a candidate design without error. Thus, we propose the first nonapproximate computational approach to generating *G*-optimal designs.

Borkowski first proposed *G*-optimal RSM designs for $K = 1, 2, 3$ experimental factors and seven experiment sizes (i.e., experimental runs denoted N) per K, generated by a genetic algorithm (GA).[4] This catalog of candidate *G*-optimal designs has evolved into a benchmark validation data set for comparing algorithm performance and resulting *G*-optimal designs over two decades of research. Rodriguez et al. applied the coordinate exchange (CEXCH) algorithm to replicate Borkowski's designs and extended it to $K = 4, 5$ factors with three run sizes each. Additionally, they compared the prediction variance properties of *G*-optimal and *I*-optimal designs and recommended *I*-optimal designs to practitioners due to lower computational cost and complexity. Saleh and Pan introduced a clustering-based CEXCH algorithm to reduce computing costs.[5] Hernandez and Nachtsheim proposed using continuous I_λ-optimal designs as a starting point for finding

highly efficient G-optimal designs, significantly reducing computational costs compared to GA and CEXCH, and extended their approach to two new $K = 4, 5$ factor design scenarios.[6] Walsh and Borkowski adapted particle swarm optimization (PSO) to the G-optimal design problem, finding it as cost-efficient as Hernandez and Nachtsheim's approach but exhibiting a high probability of generating a highly efficient design in a single algorithm run,[7] thereby mitigating the need to run the search algorithm several hundreds or thousands of times as suggested by several authors.[8] Further, in several cases, PSO produced new, improved G-optimal designs, thereby updating the catalog of candidate G-optimal designs and establishing PSO as the state-of-the-art for this problem.

The computational cost and complexity of finding G-optimal designs stems from the structure of the G-criterion, which is defined as the maximum prediction variance of the model supported by the candidate design. Calculating the G-score for a candidate design requires an optimization calculation.[1,8] The literature presents two approaches. One approach uses a gradient-based optimization method to compute the G-score, as seen in ref. [9]. However, the G-criterion function is not convex, so local optima entrapment, resulting in an incorrect G-score for the candidate design, is highly likely.[10] Miss-scoring a candidate design via this method can significantly hinder the search for the optimal design.[7,9] In the second approach, the symmetry of the prediction variance surface, for a second-order model, is exploited and a grid approach is applied to approximate the G-score of a candidate design.[4,6,7] The 5^K grid is defined as $\{-1, -0.5, 0, 0.5, 1\}^K$, and the candidate design's prediction variance is computed at each of the 5^K grid points, then the maximum of those evaluated prediction variances is returned as the candidate's G-score. Under the second-order model (RSM scenarios) this approach has been demonstrated to enable cost-efficient generation of G-optimal designs where the G-efficiencies of the resulting designs are subject to small error, typically within 1 unit on the G-efficiency scale.[7] While the grid approximation has been successful in generating optimal designs for the second-order model, it is currently unknown whether such a grid would enable G-optimal designs to search for higher-order models.

In this context, our research leverages `Gloptipoly`, a MATLAB-based implementation of the method of semi-definite relaxations of nonconvex global optimization problems, to directly compute the G-score of a candidate design without relying on grid approximations. `Gloptipoly` is capable of finding global optima of high-order polynomials, thus making it particularly suitable for computing the G-score of a candidate design as the G-criterion is equivalently viewed as the maximum of a quartic polynomial function of the experimental factors under the second-order RSM model.[2,3,11–13] By removing the grid approximation, our approach eliminates any associated errors in computing a candidate's G-score, leading to more accurate and reliable optimal designs. Our approach not only improves the precision of G-optimal design generation but also opens new possibilities for extending the search to higher-order RSM models. Through comprehensive case studies, we demonstrate the efficacy of our approach and compare results to those previously published.

We present our notation and formulation of the G-optimal design problem in Section 2, and illustrate that the G-score is a polynomial function of the experimental factors. In Section 3 we provide proof of concept that `Gloptipoly` can calculate the G-score of a candidate design without error by applying it to score the current best-known G-optimal RSM designs for $K = 1$ to 5 experimental factors as in Walsh and Borkowski.[7] In Section 4 we explore two algorithms, the Nelder–Mead simplex (NMS) algorithm and PSO, to search the space of candidate matrices while utilizing `Gloptipoly` to score candidate designs suggested by these algorithms. In Section 5 we extend this methodology to generating G-optimal designs for models beyond the second-order polynomial. Last, we provide concluding remarks and suggestions for future research in Section 6.

2 | G-OPTIMAL DESIGN FOR RSM SCENARIOS

2.1 | Small exact response surface designs

We consider the second-order linear model under standard assumptions. Let N represent the number of design points (i.e., number of affordable experimental runs) and K represent the number of experimental factors.[1] The row-vector $\mathbf{x}' : 1 \times K$ represents a design point (i.e., single experimental run on K factors). Experimental factors are assumed scaled to range $[-1, 1]$ and so the design space is the $\mathcal{X} = [-1, 1]^K$ hypercube.[1,8] The goal of optimal experimental design is to distribute the N design points $\mathbf{x}_i'$, $i = 1, \ldots, N$ over $\mathcal{X}$ in some optimal configuration.

Matrix $\mathbf{X}$ of dimension $N \times K$ represents the exact (as opposed to continuous) design matrix. The result of an optimal design search is to populate this matrix with factor settings that are optimal under the experimenter's objectives. Therefore, to find the optimal design, one must search over the space of candidate matrices which is a NK-dimensional hypercube,[7]

that is,

$$\mathbf{X} \in \bigtimes_{j=1}^{N} \mathcal{X} = \bigtimes_{j=1}^{N} [-1,1]^K = [-1,1]^{NK} = \mathcal{X}^N. \tag{1}$$

We assume the second-order linear model which has $p = \binom{K+2}{2}$ parameters. This model is commonly written in scalar form as

$$y = \beta_0 + \sum_{i=1}^{K} \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \beta_{ij} x_i x_j + \sum_{i=1}^{K} \beta_{ii} x_i^2 + \epsilon.$$

Matrix $\mathbf{F}(\mathbf{X})$ of dimension $N \times p$ represents the model matrix with rows given by the expansion vector $\mathbf{f}'(\mathbf{x}_i') = (1 \quad x_{i1} \quad \dots \quad x_{i2} \quad x_{i1}x_{i2} \quad \dots \quad x_{i(K-1)}x_{iK} \quad x_{i1}^2 \quad \dots \quad x_{iK}^2)$. We abbreviate the model matrix as $\mathbf{F}$, with the understanding that it is always a function of the design matrix $\mathbf{X}$.[7] Then, the response surface model becomes a linear model of $\mathbf{y} = \mathbf{F}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where we assume $\boldsymbol{\epsilon} \sim \mathcal{N}_N(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ and $\mathcal{N}_N$ denotes the N-dimensional multivariate normal distribution. The ordinary least squares estimator of $\boldsymbol{\beta}$ is $\hat{\boldsymbol{\beta}} = (\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'\mathbf{y}$ which has variance $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{F}'\mathbf{F})^{-1}$. The total information matrix for $\boldsymbol{\beta}$, specifically $\mathbf{M}(\mathbf{X}) = \mathbf{F}'\mathbf{F}$, plays an important role in the optimal design of experiments—all optimal design objective functions are functions of this matrix.[8]

The quality of a candidate design $\mathbf{X} \in \mathcal{X}^N$ is measured by an optimality criterion defined by the practitioner. The space of candidate matrices $\mathcal{X}^N$ is explored via an additional optimization algorithm in an effort to find the "'best," or optimal, design. Thus, an exact optimal design problem is defined by three components[7]: (1) the number of design points N that can be afforded in the experiment, (2) the structure of the model one wishes to fit (here the second-order model), and (3) the criterion which defines an optimal design, as a function of $\mathbf{M}(\mathbf{X})$. In the next subsection, we define the G-criterion, the G-optimal design, and several efficiency measures commonly used to evaluate candidate designs.

2.2 | *G*-Optimal design

The G-optimal design minimizes the model's maximum prediction variance (i.e., the worst-case scenario) over design space $\mathcal{X}$. At any point of prediction $\mathbf{x}' \in \mathcal{X}$ the model's mean prediction variance is

$$\mathrm{Var}(\hat{y}(\mathbf{x}')) = \sigma^2 \mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}').$$

The scaled prediction variance (SPV) removes the scale parameter σ^2 and re-scales to N by multiplying by the factor N/σ^2.[7] Thus, for candidate design $\mathbf{X}$, SPV is defined as

$$\mathrm{SPV}(\mathbf{x}'|\mathbf{X}) := N\mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}'). \tag{2}$$

The G-score of a candidate design $\mathbf{X}$ is the maximum scaled prediction variance over all points of prediction $\mathbf{x}' \in \mathcal{X}$

$$G(\mathbf{X}) := \max_{\mathbf{x}' \in \mathcal{X}} \mathrm{SPV}(\mathbf{x}'|\mathbf{X}). \tag{3}$$

Equation 3, thus, shows that for a fixed candidate design $\mathbf{X}$, scoring the candidate on the G optimal criterion is itself an optimization problem. The G-optimal design matrix $\mathbf{X}^*$ is the matrix that minimizes, overall design matrices $\mathbf{X} \in \mathcal{X}^N$, the maximum SPV

$$\begin{aligned} \mathbf{X}^* &:= \operatorname*{argmin}_{\mathbf{X} \in \mathcal{X}^N} G(\mathbf{X}) \\ &= \operatorname*{argmin}_{\mathbf{X} \in \mathcal{X}^N} \max_{\mathbf{x}' \in \mathcal{X}} \mathrm{SPV}(\mathbf{x}'|\mathbf{X}). \end{aligned} \tag{4}$$

Equation 4 shows that finding the G-optimal design is a minimax problem. This optimization has proved notoriously difficult to solve because neither of the inner and outer optimizations required to compute $\mathbf{X}^*$ are convex in large part due to the expansion of the design matrix into model matrix $\mathbf{F}$.[4,6,9]

The General Equivalence Theorem of refs. [14–16] demonstrates that the lower bound on $G(\mathbf{X})$ is

$$G(\mathbf{X}) = \max_{\mathbf{x}' \in \mathcal{X}} \text{SPV}(\mathbf{x}') \geq p. \tag{5}$$

That is, the smallest value that the maximum scaled prediction variance of a candidate design $\mathbf{X}$ can be is p, the number of parameters. This yields a way to verify if a proposed exact G-optimal design is globally optimal; however, not all design scenarios have the globally G-optimal designs that will achieve this lower bound. Further, if design $\mathbf{X}^*$ is globally G-optimal and achieves the result in Equation 5, then for this design $\text{SPV}(\mathbf{x}'|\mathbf{X}) = p$ at all diagonals of the hat matrix $\mathbf{F}(\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'$ and $\text{SPV}(\mathbf{x}'|\mathbf{X}) \leq p$ at all other points of prediction $\mathbf{x}' \in \mathcal{X}$.[1] It is customary to exploit this fact and score candidate designs on the G-efficiency scale,

$$G_{\text{eff}}(\mathbf{X}) = 100\frac{p}{G(\mathbf{X})}, \tag{6}$$

in order to gauge the quality of a candidate design $\mathbf{X}$ (larger G_{eff} on this scale implies a better design). Lastly, the *relative efficiency* of two candidate designs may be computed as

$$G_{\text{releff}}(\mathbf{X}_1, \mathbf{X}_2) = 100\frac{G_{\text{eff}}(\mathbf{X}_1)}{G_{\text{eff}}(\mathbf{X}_2)}. \tag{7}$$

2.3 | Previously studied design scenarios

In Table 1 we provide a description of all exact G-optimal design scenarios discussed in the literature to date by authors.[4–7,9] We will study several of these design scenarios via the application of `Gloptipoly` in subsequent sections.

2.4 | SPV as a polynomial function of the experimental factors

In this section, we provide a simple $K = 1, N = 3$ mathematical example to illustrate that the SPV function of a second-order linear model is a quartic polynomial. Generally, denote the corresponding candidate design as

$$\mathbf{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

where we have one-factor x with levels a, b, c. The corresponding model matrix (under the second-order RSM model) is

$$\mathbf{F} = \begin{bmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{bmatrix}.$$

From this model matrix, (and without loss of generality) denote the inverse of the model-information matrix as

$$(\mathbf{F}'\mathbf{F})^{-1} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}.$$

Then we may use the inverse information matrix to compute the SPV under design $\mathbf{X}$ at arbitrary design point $x \in [-1, 1]$ as

TABLE 1 Summary of design scenarios, algorithms, and authors who have addressed the exact G-optimal design problem for the second-order RSM model in the last 20 years.

K : Number of experimental factors	N : Number of experimental runs	Algorithm	Authors
1	3, 4, 5, 6, 7, 8, 9	GA	Borkowski
		$G(I_\lambda)$-CEXCH	Hernandez and Nachtsheim
		PSO	Walsh and Borkowski
2	6, 7, 8, 9, 10, 11, 12	GA	Borkowski
		CEXCH	Rodriquez et al.
		cCEA ($N = 7$ to 12)	Saleh and Pan
		PSO	Walsh and Borkowski
3	10, 11, 12, 13, 14, 15, 16	GA	Borkowski
		CEXCH	Rodriquez et al.
		cCEA ($N = 11$ to 16)	Saleh and Pan
		PSO	Walsh and Borkowski
4	15, 20, 24	CEXCH	Rodriguez et al.
		cCEA ($N = 24$)	Saleh and Pan
		PSO	Walsh and Borkowski
	16	cCEA	Saleh and Pan
	17	$G(I_\lambda)$-CEXCH	Hernandez and Nachtsheim
		PSO	Walsh and Borkowski
5	21, 26, 30	CEXCH	Rodriquez et al.
		cCEA ($N = 26$)	Saleh and Pan
		PSO	Walsh and Borkowski
	23	$G(I_\lambda)$-CEXCH	Hernandez and Nachtsheim
		PSO	Walsh and Borkowski

Abbreviations: CEXCH, coordinate exchange; GA, genetic algorithm; PSO, particle swarm optimization; RSM, response surface methodology.

$$SPV(\mathbf{x}'|\mathbf{X}) = 3 \times \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}.$$

$$= 3 \times \begin{bmatrix} f_{11} + f_{21}x + f_{31}x^2 & f_{12} + f_{22}x + f_{32}x^2 & f_{13} + f_{23}x + f_{33}x^2 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

$$= 3f_{11} + x3(f_{21} + f_{12}) + x^2 3(f_{31} + f_{22} + f_{13}) + x^3 3(f_{32} + f_{23}) + x^4 3(f_{33})$$

which is a fourth-order polynomial. Thus, computing the G-score of a candidate design can be formulated as optimizing a polynomial function of the experimental factors. The polynomial coefficients (i.e., the f_{ij}-terms) are functions of the entries in the candidate design $\mathbf{X}$, and the optimization is over the arbitrary point $x \in [-1, 1]$. A similar derivation for any $K \geq 2$ is straightforward.

To visualize the problem, we consider the $K = 2$, $N = 9$ design scenario. Figure 1 contains contour plots of the SPV surface for the G-optimal design reported by Walsh and Borkowski[7] (the right panel) and a random design generated under a multivariate uniform distribution (the left panel) (note that the SPV for this contour plot is plotted on the log scale). The red-yellow-filled points on the graphic represent the 5^K grid over which SPV is computed to approximate the maximum. The large red "X"s indicate the true location of the maximum prediction variance.

The quartic polynomial structure over two factors for the G-optimal design (left panel) is readily apparent. Further, one can see that while SPV at grid point $(0, -1)$ is reported as this design G-score, the true maximum SPV is slightly to the left of this point, illustrating a small error in approximating this design G-score. In the right panel, it is readily apparent that the

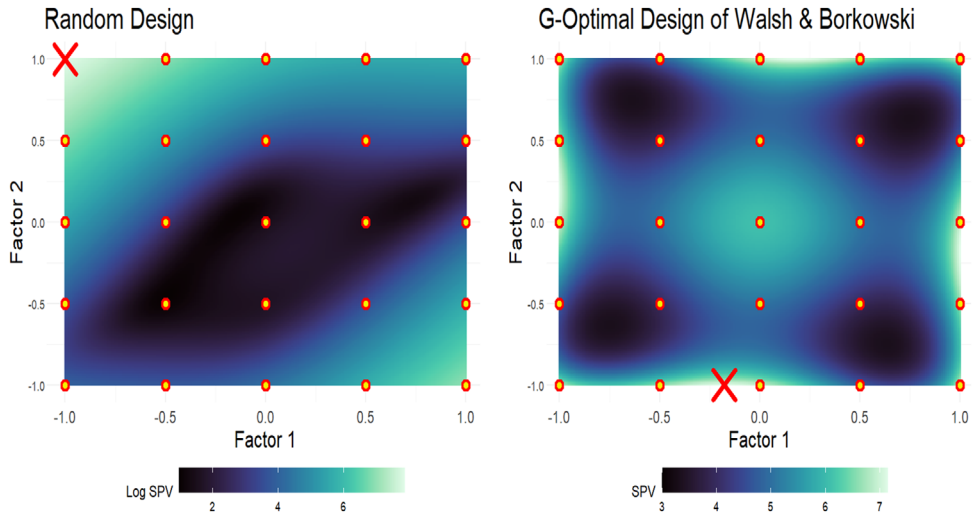


FIGURE 1 Side-by-side SPV-surface plots for a completely random design (left) and for the *G*-optimal design of ref. [7] (right). Locations that would be sampled by the grid approximation, as in the literature,[4,7] are given as yellow circles outlined in red. SPV, scaled prediction variance.

SPV surface contains a much larger prediction variance over the design space than the *G*-optimal design (the color in this graphic is on the log scale). Nonetheless one can still see the quartic polynomial structure. Further, the grid approximation returns the SPV at point $(-1, 1)$ and the true SPV also occurs at this location, leading to no error in computing the *G*-score of this candidate design.

Therefore, computing the *G*-score of a candidate design is precisely the optimization problem solved by `Gloptipoly`. In the next section, we adapt `Gloptipoly` to computing a design *G*-efficiency and score all best-known *G*-optimal designs reported in Walsh and Borkowski[7] to (1) act as a proof-of-concept and validate that `Gloptipoly` can correctly score these designs and (2) elucidate the size of error in the *G*-efficiency approximations for each design that results from using the 5^K grid approximation.

3 | PROOF-OF-CONCEPT: SCORING *G*-OPTIMAL DESIGNS VIA GLOPTIPOLY

3.1 | The generalized problem of moments: brief review of `Gloptipoly`'s development

`Gloptipoly` was developed for solving, and in more complex situations, approximating the generalized problem of moments (GPM).[11–13] GPM is a linear optimization problem with countably many constraints on the space of measures[13] and of which global polynomial optimization is a special case.[17] For a complete discussion of methodologies and applications, see.[18]

Algorithmic development for solving GPM has been a central area of focus in the optimization research community since the late 20th century. Hernandez-Lerma and Lasserre introduced a method of using approximation schemes to address infinite linear programs.[19] Their work demonstrates that under certain assumptions, optimal solutions to a GPM can be approximated by finding the optimal solution to each of a sequence of finite-dimensional linear programs associated with the original problem. These are sometimes called linear program (LP) relaxations.[19] Their technique takes an infinite-dimensional linear program and discretizes it, breaking the problem up into a sequence of finite-dimensional linear programs of increasing size. They demonstrate that an accumulation point for a sequence of optimal solutions to one of the finite-dimensional approximating programs is also an optimal solution for the original infinite-dimensional program. This powerful result informed the theoretical development in ref. [11], which expanded the idea of finding global polynomial optima by reducing the problem of moments to a sequence of convex linear matrix inequality (LMI) problems. There are, however, some issues with using LP relaxations to solve the GPM. Numerical instability and the absence of a general convergence guarantee make the LP relaxation technique inappropriate for finding optima of less well-behaved functions. To overcome these challenges, semidefinite program (SDP) relaxations were introduced.[12] SDP relaxations offer an attractive solution for small-scale problems because they include a convergence guarantee and resolve some numerical

stability issues, but they have not yet been implemented to solve very large optimization problems. Note that the dimension of optimization problems we are attempting to solve, via `Gloptipoly`, has $K = 1, 2, 3, 4, 5$. Therefore, the currently available tools are ideal for our application here.

It should be noted that `Gloptipoly` is not a standalone GPM solver; rather, it converts the original problem into a sequence of SDPs that are solved by calling an external semidefinite solver. The program `SeDuMi`[20] is the default solver in `Gloptipoly` and appears to be widely used in SDPs. Note that the software is configured to work with any other semidefinite solver in `YALIMP`, a MATLAB toolbox for interfacing SDP solvers.[21]

3.2 | Computing a candidate designs's *G*-score via `Gloptipoly`

We first validate `Gloptipoly` as an appropriate tool to accurately compute the G-score of a candidate design by adapting it to this problem and computing the G-efficiencies of the current best G-optimal designs for $K = 1, 2, 3, 4, 5$ experimental factors reported in Walsh and Borkowski.[7] While constituting the set of best-know designs for RSM scenarios, it is already known that the G-scores of these designs are subject to small error.[7] Table 2 contains G-efficiencies, computed as in Equation 6. All MATLAB code used to generate the results of this paper, as well as instructions on how to implement a reproducible example, can be found on the author's Github page: https://github.com/HyrumHansen/thesisResearch.

Table 2 contains, for each design scenario, (1) the best known G-optimal design's G-efficiency computed under the 5^K grid approximation (2) the corresponding G-efficiency computed on the same design via `Gloptipoly`, and (3) the absolute difference of the two scores (i.e., the error in approximating the G-efficiency). One can see that for the majority of design scenarios, the error in G-efficiency calculation is small, under 1% on the G-efficiency scale. Three of the design scenarios show an appreciably larger G-score approximation error:

1. $K = 2, N = 9$, error = 2.60 efficiency units,
2. $K = 2, N = 10$, error = 1.10 efficiency units,
3. $K = 3, N = 10$, error = 1.05 efficiency units.

This suggests some room for improvement as these designs' true G-efficiencies have been optimistically, and incorrectly, scored. To verify this, we also scored these designs using a very dense grid $\mathbf{x}' \in \{-1, -0.99, -0.98, \dots, 0.98, 0.99, 1\}^K$. We found the scores reported by `Gloptipoly` to be in strong agreement with those reported by the dense grid SPV evaluations, specifically, all case errors were (± 0.01) on the G-efficiency scale.[22]

Thus, we have demonstrated that `Gloptipoly` is a fit-for-purpose tool for accurately computing G-scores of candidate designs. Having established this fact, we now extend to searching the space of candidate matrices for the G-optimal design while using `Gloptipoly` as a component to score candidate designs. We discuss this in the next section.

4 | EXPLOITING `GLOPTIPOLY` TO AID THE *G*-OPTIMAL DESIGN SEARCH

So far we have demonstrated that `Gloptipoly` can accurately solve the problem of computing a candidate design's G-score, that is, `Gloptipoly` can be trusted to solve Equation 3. The search for the G-optimal design is an additional optimization process over the space of candidate matrices as expressed in Equation 4, that is, we need an additional optimizer to solve

$$\mathbf{X}^* = \underset{\mathbf{X} \in \mathcal{X}^N}{\operatorname{argmin}}\, G(\mathbf{X})$$

where, at each iteration of the optimization, `Gloptipoly` will be called on each candidate design fed to it by this optimizer.

The task now is to learn the efficacy and computational cost of this approach. We implemented studies of two optimization algorithms, both available in MATLAB, to search the space of candidate matrices:

1. NMS algorithm, and
2. particle swarm optimization.

NMS was chosen for this study due to its ease of implementation and ability to enforce design space constraints. In addition, it is a gradient-free method, thus having the global search capability (i.e., it is somewhat robust to entrapment at

TABLE 2 Relative efficiencies as reported by the grid approximation and Gloptipoly for all second-order design scenarios covered by ref. [7].

K : Number of experimental factors	N : Number of experimental runs	G- efficiency via 5^K grid approx.	G- efficiency via `Gloptipoly`	Absolute difference
1	3	100	100	0
	4	82.92	82.92	0
	5	80.58	80.58	0
	6	100	100	0
	7	91.17	91.17	0
	8	89.13	89.13	0
	9	100	100	0
2	6	75.03	74.39	0.64
	7	80.24	80.04	0.19
	8	87.94	87.94	0
	9	86.63	84.03	2.60
	10	87.40	86.30	1.10
	11	87.07	86.66	0.41
	12	88.17	88.11	0.06
3	10	71.43	70.38	1.05
	11	80.51	79.54	0.97
	12	83.35	83.12	0.22
	13	86.46	85.81	0.64
	14	89.71	89.09	0.61
	15	85.99	85.77	0.22
	16	85.79	85.39	0.39
4	15	71.09	70.64	0.44
	17	73.90	73.66	0.24
	20	80.20	79.31	0.89
	24	85.95	85.85	0.10
5	21	68.67	67.84	0.83
	23	73.19	72.67	0.52
	26	75.31	74.84	0.47
	30	76.16	75.71	0.45

Note: The grid approximation tends to slightly over-approximate the G-efficiency for a candidate design when compared to `Gloptipoly`, but the difference is generally marginal. For some cases, the $K = 2$, $N = 9$ design scenario being a notable example, the difference sufficiently pronounced to consider switching algorithms.

local(sub)-optimal solutions). PSO was chosen because it is the current state-of-the-art for the G-optimal design problem and MATLAB contains a library with the canonical implementations of PSO.

For the performance assessment of the NMS-`Gloptipoly` pairing, we implemented this approach to all $K = 1, 2, 3$ experimental factor cases explored in Borkowski,[4] which constitute seven experimental runs per K for a total of 21 design scenarios. This set of problems has the greatest collection of exploration and evidence. For example, Hernandez and Nachtsheim provided a comprehensive comparison of the performance of GA, the standard G-CEXCH,[9] and their proposed $G(I_\lambda)$-CEXCH[6] in which they ran each algorithm 200 times per design scenario and published the efficacy of $G(I_\lambda)$-CEXCH and G-CEXCH relative to designs produced by the GA. Further, Walsh and Borkowski[7] extended this comparison to include an assessment of G-PSO's efficacy. For our study, we ran NMS-`Gloptipoly` 500 times. In the next section, we update the assessment of all algorithms relative to GA, including our results. All $500 \times 21 = 10500$ optimization searches were run in parallel on an 18-core i9 CPU, and the entire study took approximately one week to compute. The implementation of `Gloptipoly` as an optimizer *for every candidate design at every iteration of NMS* no doubt adds to computing costs. However, all codes were run in MATLAB (uncompiled), so the computing cost was also confounded with the

TABLE 3 Relative efficiencies of G-optimal designs for each algorithm relative to GA generated designs of ref. [4].

Design scenario		Best design efficiency relative to G-GA			
K	N	G-PSO	$G(I_\lambda)$-CEXCH	G-CEXCH	G-NMS/Gloptipoly
1	3	100.0	100.0	100.0	100.0
	4	100.0	96.2	98.7	100.0
	5	100.0	97.0	98.7	100.0
	6	100.0	100.0	100.0	100.0
	7	100.0	98.8	99.7	100.0
	8	100.0	94.7	99.4	100.0
	9	100.0	100.0	89.4	100.0
2	6	100.3	94.1	96.5	99.0
	7	100.1	95.5	97.9	98.6
	8	100.0	94.7	99.7	99.2
	9	100.3	95.8	97.0	97.8
	10	101.7	93.2	97.5	100.3
	11	101.0	97.0	94.0	100.1
	12	103.9	95.1	101.2	100.3
3	10	101.6	95.4	93.1	94.3
	11	104.2	96.9	92.9	97.9
	12	103.8	90.3	90.7	99.6
	13	103.2	99.9	92.9	94.3
	14	100.5	100.0	87.6	92.0
	15	102.5	100.1	98.5	94.4
	16	108.1	100.2	100.1	97.4

Note: From left to right, efficiencies for the PSO of ref. [7], the $G(I_\lambda)$ procedure of ref. [23] the coordinate-exchange of ref. [9], and the Nelder–Mead + `Gloptipoly` approach studied in this work.
Abbreviations: CEXCH, coordinate exchange; GA, genetic algorithm; NMS, Nelder–Mead simplex; PSO, particle swarm optimization.

choice of computing language. Further, `Gloptipoly`'s structure is quite different from typical optimization algorithms. We attempted to retrieve a cost metric such as a number of iterations, but were unsuccessful in identifying a suitable analog for this quantity in the `Gloptipoly` output. Therefore, we only report a summary of the best designs found for each algorithm over their number of runs, n_{run}.

4.1 | Results

4.1.1 | NMS-`Gloptipoly` results

Boxplots for all 500 runs per design scenario in Figure 2 are given to emphasize the prevalence of local optima. Naturally, lower-dimensional problems have fewer local optima and the algorithm generally finds designs with high G-efficiency, but as the dimension of the problem increases so does the number of local optima and hence the algorithm's propensity to entrapment. A comparison of the best designs found by $G-$NMS/Gloptipoly to G-PSO,[7] $G(I_\lambda)-CEXCH$,[6] the standard CEXCH[6] and G-GA[4] is provided in Table 3. $G-$NMS/Gloptipoly appears to be competitive with all algorithms but PSO.

4.1.2 | PSO-`Gloptipoly` results

Figure 3 contains boxplot comparisons of the best solutions found by NMS-`Gloptiploly` ($n_{\text{run}} = 500$) and PSO-`Gloptipoly` ($n_{\text{run}} = 20$). In only 4% of the number of algorithm runs, PSO returns a better design than Nelder–Mead. Further, in several cases, the PSO-`Gloptipoly` has produced better designs than those reported in ref. [7]. These results (1) underscore that PSO is an excellent match with the structure of the G-criteria and (2) while the current MATLAB

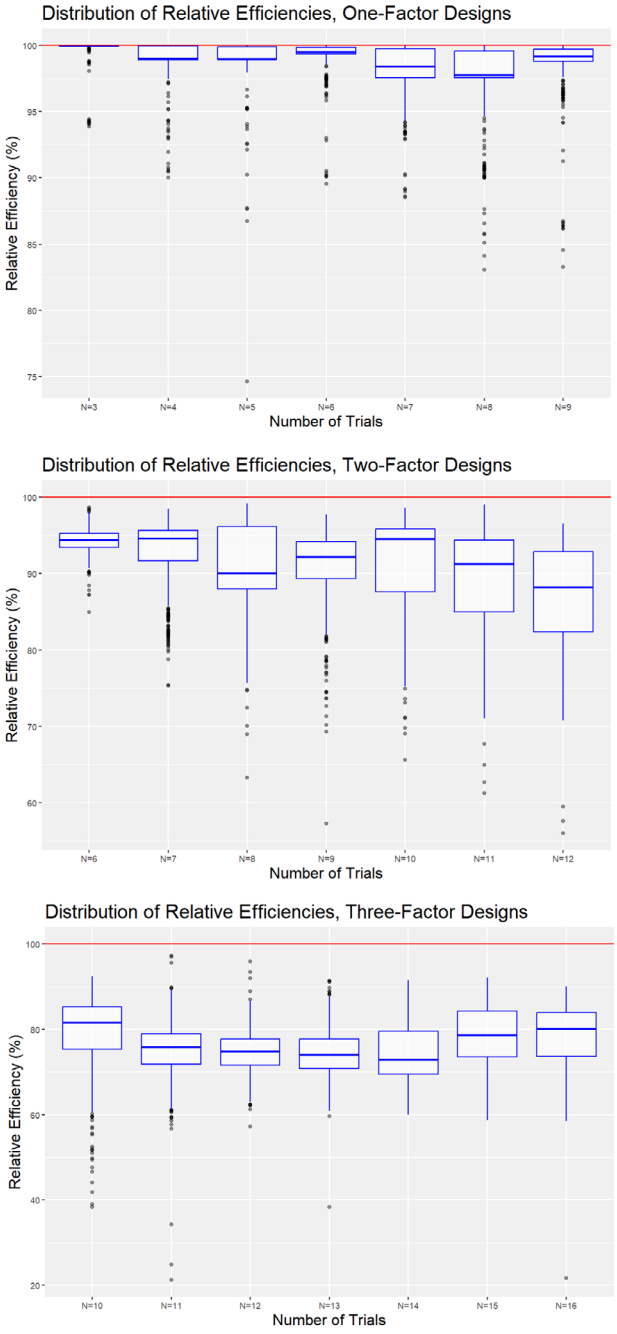


FIGURE 2 The top panel: Boxplots for 500 runs of Nelder–Mead on one-factor design scenarios. The best of these runs is consistently the *G*-optimal design for one-factor design scenarios. The middle panel: Boxlpots for two-factor settings. A two-factor design search results in significantly more spread when compared to the one-factor design search, likely due to an increase in the number of local optima. The bottom panel: Boxplots for three-factor settings. The dimension of the problem starts to challenge NMS, however, the best found designs all have relative efficiencies above 90%. NMS, Nelder–Mead simplex.

implementation does take nonnegligible computer time, the PSO-`Gloptipoly` approach has shown the ability to find highly efficient *G*-optimal designs in a single run with large probability.

5 | GENERATING *G*-OPTIMAL DESIGNS BEYOND THE SECOND-ORDER RSM MODEL

The order of the polynomial model being designed is a fundamental assumption in RSM, while the true nature of the response surface is usually unknown. The second-order model is often assumed (especially in practice) as the

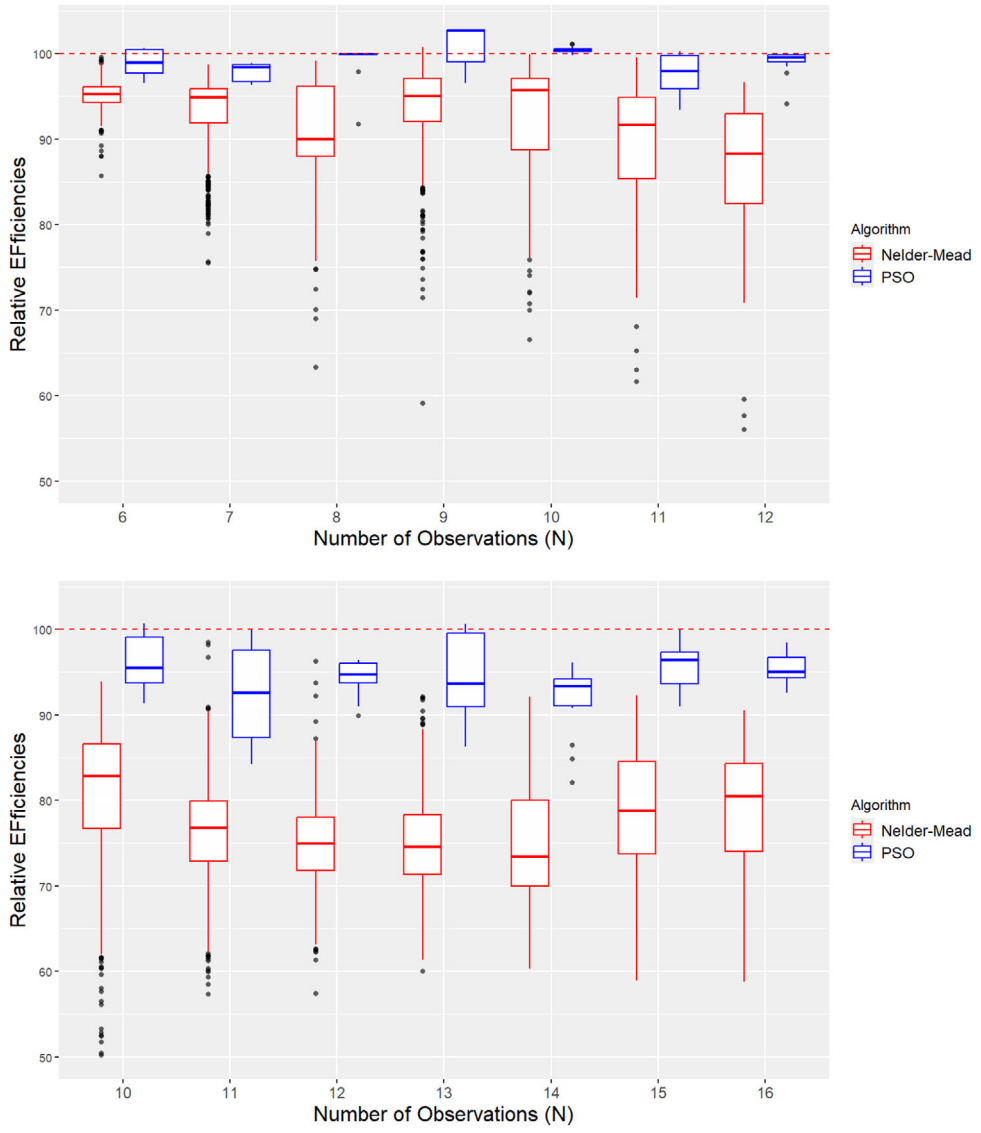


FIGURE 3 Boxplots for the distribution of designs by design scenario and algorithm, with efficiencies relative to the best designs of ref. [7] for all two factor settings (top panel), and all three factor settings(bottom panel). The best design of just 20-runs of PSO surpassed the best design of 500 runs of Nelder–Mead in nearly every case. PSO, particle swarm optimization.

second-order polynomial (with interactions) provides a fair amount of flexibility in fitting a response surface and can model a local optimum. In some scenarios, the response surface is more complex than can be adequately modeled by a second-order polynomial. In such cases, higher-order polynomials, such as cubic or quartic, may be necessary to capture the true nature of the response surface.[1] As Box and Draper point out: "There are situations where the second-order model is insufficient to describe the true nature of the system, especially when the response surface exhibits multiple inflection points or complex curvatures. In such cases, fitting higher-order polynomial models provides a significantly better representation of the experimental data and improves the accuracy of predictions".[24] Thus, higher-order models become essential when the system under study exhibits complex interactions and nonlinearities that a second-order model cannot capture. Such models provide a more accurate fit to the data, especially in processes where precision in optimization is critical.[1,25] Moreover, Dean et al. emphasized the practical importance of higher-order models, asserting that the use of higher-order models is crucial in scenarios where the experimental region is highly nonlinear.[26] These models improve the ability to explore and optimize the response surface, leading to better decision-making in experimental design.[26] These insights collectively underscore the importance of higher-order models in accurately capturing and optimizing complex response surfaces in experimental studies.

We are currently unaware of any approach that enables G-optimal design for higher-order RSM models. Our approach, via `Gloptipoly` to calculate the G-score for a candidate design, solves this problem. In the next sections, we illustrate proof-of-concept and explore the G-optimal designs for several higher-order model design scenarios.

5.1 | Models with higher-order interaction terms

We first consider a $K = 2$ experimental factor scenario with two run sizes, $N = 9, 10$ under a second-order model with interactions between main-effects and quadratic terms (thus adding additional flexibility to the parabolic surface that can be fit to the experimental data). Specifically, we assume that the true response surface should be modeled as

$$f(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1^2 x_2 + \beta_6 x_1 x_2^2$$

and we seek a G-optimal design to support this model.

We also apply our approach to $K = 3$ factor scenario with $N = 14, 15$ runs under a similar assumed response surface model:

$$\begin{aligned} f(x_1, x_2, x_3) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 \\ + \beta_4 x_1^2 + \beta_5 x_2^2 + \beta_6 x_3^2 + \beta_7 x_1 x_2^2 + \beta_8 x_1^2 x_2 + \beta_9 x_2 x_3^2. \end{aligned}$$

Results will be discussed in subsequent sections.

5.2 | Cubic and quartic models

We explore four design scenarios under the cubic model. First $K = 1$ factor with run sizes $N = 5, 6$ under model

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$

And $K = 2$ factor experiments with run sizes $N = 9, 10$ under model

$$f(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1^3 + \beta_6 x_2^3.$$

Lastly, we investigate two $K = 2$ experimental factors with run sizes $N = 11, 12$ under the quartic polynomial model

$$f(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1^3 + \beta_6 x_2^3 + \beta_7 x_1^4 + \beta_8 x_2^4.$$

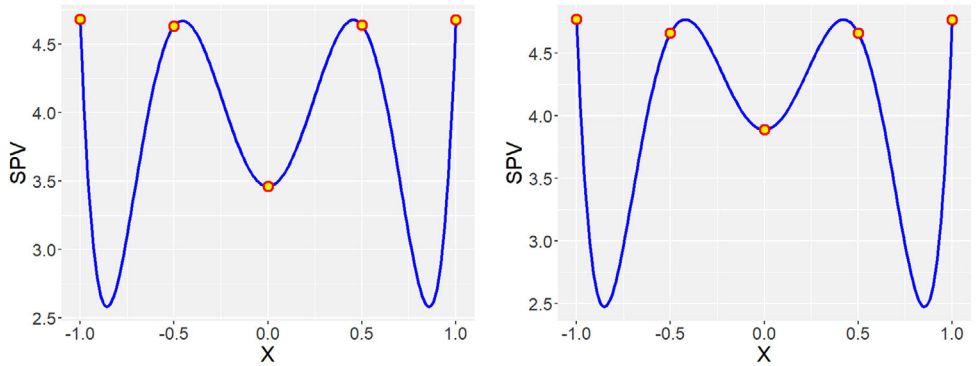
5.3 | Results

We ran NMS-`Gloptipoly` approach $n_{\text{run}} = 100$ times on each design scenario. The distribution of run-times on our machine is reported in Table 4. The $K = 3$ scenarios show a significant increase in run time (i.e., by a factor of 10) over the $K = 2$ factor scenarios.

We now consider the best G-optimal designs found from the searches under the cubic model. In Figure 4 we visualize the SPV surfaces for optimal designs produced for these same one-factor settings. The grid approximation points of ref. [4] are presented as yellow dots with red outlines. In both cases, the max SPV is achieved when $x = 1$, so the grid approximation works just as well as `Gloptipoly` for the univariate case. Figure 5 contains plots of the G-optimal designs and corresponding SPV surfaces for $K = 2$, $N = 9, 10$ design scenarios for (1) quadratic model, (2) quadratic model plus interaction, and (3) cubic model quadratic for comparison. Designing for a second-order model with interaction alters both the orientation of design points and the structure of the SPV surface of the G-optimal design in a nonnegligible way. The SPV plots of the design under the cubic model show that SPV is a sixth-order polynomial. The structure of the SPV surface for both nearly $G-$optimal designs indicates that the 5^K grid may be effective in helping generate approximate G-optimal designs for cubic polynomial models, evidenced by the maximum SPV occurring close to one of the grid points for each scenario.

TABLE 4 Table of run-times (minutes) for selected design-scenarios for higher-order models.

Model	K	N	Min	Median	Max
Higher-order interactions	2	9	25.05	25.28	25.84
	2	10	23.78	23.86	25.13
	3	14	192.92	194.42	194.68
	3	15	192.72	193.88	194.68
Cubic	1	5	2.30	2.46	3.15
	1	6	4.08	4.11	4.12
	2	9	27.04	27.74	27.95
	2	10	28.64	28.67	29.00
Quartic	2	11	21.49	21.77	22.03
	2	12	24.54	24.56	24.84



(A) SPV surface for $K = 1, N = 5$ (B) SPV surface for $K = 1, N = 6$

FIGURE 4 *SPV*-surfaces for the 5-run (left) and 6-run (right) best G-optimal designs found under the cubic model for $K = 1$ factor. The yellow points on these SPV surfaces represent the 5^K grid used by several authors to generate G-optimal designs for the second-order model. These graphics suggest that the 5^K grid may also be effective for generating G-optimal designs for cubic models, evidenced by the maximum SPV occurring at one of the grid points. SPV, scaled prediction variance.

We now consider the best G-optimal designs found from the searches under the quartic model. In Figure 6 we provide the SPV plot for the most G-efficient design proposed by our algorithm for $K = 11, 12$, respectively, with design-points superimposed. Given the complexity of the SPV structure, an eighth-order polynomial is apparent. The symmetry in the location of inflection points on these SPV surfaces suggests that the 5^K grid may be useful to approximate G-optimal design generation for fourth-order RSM models.

6 | CONCLUSIONS

Research on G-optimal design has been quite active in the last two decades and our community continues to generate new insights into effectively solving this problem. To this point, generating G-optimal RSM models has remained a challenging computational task. In this paper we addressed a core component of the difficulty of this problem – using `Gloptipoly`, one can, without approximation, accurately compute the G-score for a candidate design under any polynomial model. The use of `Gloptipoly` as a component of a search algorithm for G-optimal designs thus enables the first nonapproximate computational method and opens new research possibilities for generating G-optimal designs for higher-order RSM models.

We illustrated the correctness and efficacy of our approach on a wide set of design scenarios published in the literature. We benchmarked the performance of the NMS algorithm and particle swarm optimization on a large set of previously studied design scenarios. Nelder–Mead with `Gloptipoly` performed reasonably well in finding highly efficient G-optimal designs for $K = 1, 2$ factor scenarios, but its efficacy decreased as the dimension of the problem increased, and the only solution is to run the algorithm more times in higher dimensional scenarios. Although more computationally expensive,

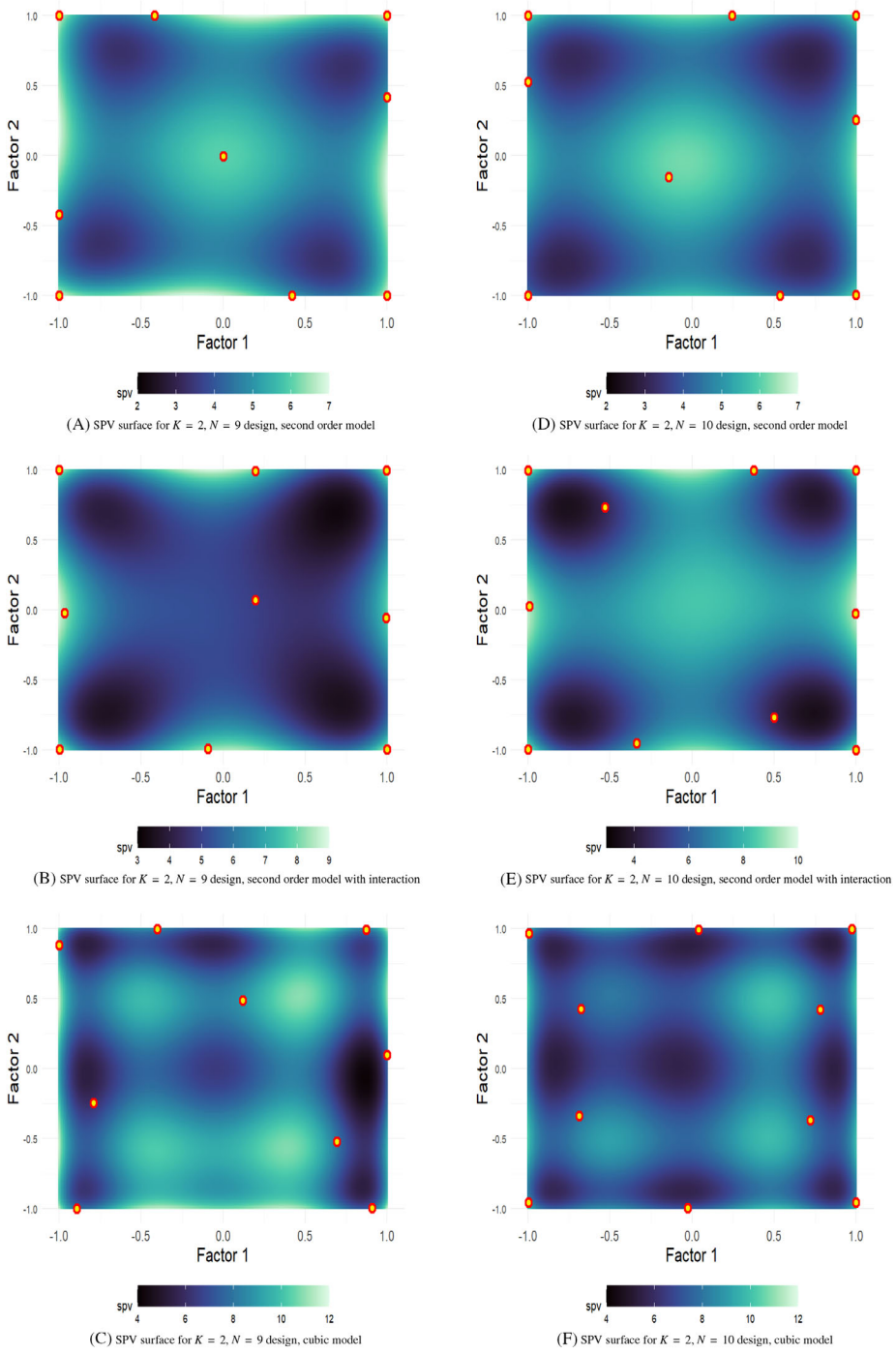


FIGURE 5 The yellow points on these graphs represent the *G*-optimal design points and illustrate how the cubic model affects design geometry and structure of the SPV surface. (left column) SPV surface and the generated *G*-optimal design for (A) second-order model, (B) second-order model with interaction term, and (C) third-order model. (right column) SPV surface and the generated *G*-optimal design for (A) second-order model, (B) second-order model with interaction term, and (C) third-order model. SPV, scaled prediction variance.

the PSO/`Gloptipoly` approach was able to not only reproduce the current best *G*-optimal designs,[7] but also in several cases marginally improve on these designs in just 20 runs of the algorithm. These improvements were due in large part to the correction of the error introduced by using the 5^K grid approximation used in previous studies.[4,6,7]

We also studied several design scenarios under higher-order models and were able to generate highly *G*-efficient designs. This proof-of-concept opens new avenues for research on this problem. All designs generated in this study can be found on the author's Github site.

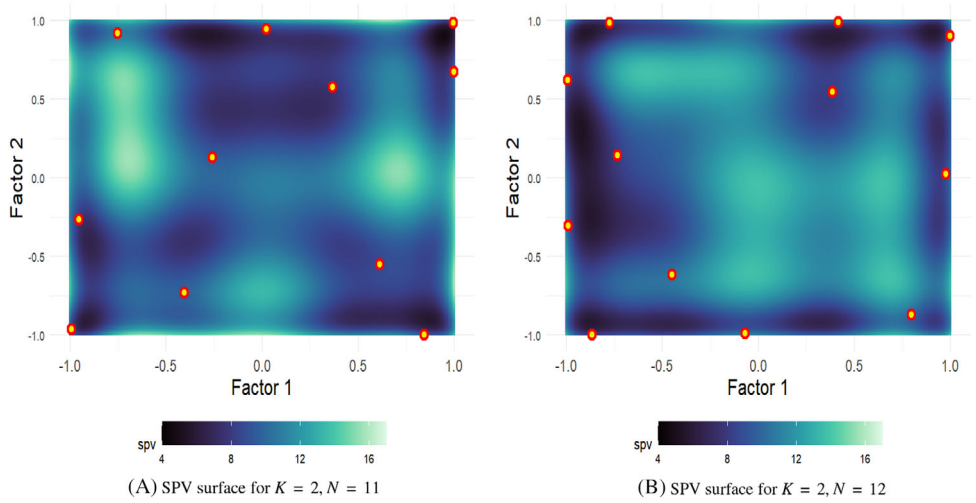


FIGURE 6 SPV surfaces for the quartic model in two-factors for 11 (left) and 12 (right) trials. There is no clear design-point symmetry for either of these surfaces. SPV, scaled prediction variance.

The computational cost of our approach is somewhat expensive, which may be still tolerable for academic research but perhaps not for practical applications. Currently, MATLAB is the only language that we found with an implementation of `Gloptipoly`. Future research may include developing the algorithm in a compiled language in order to reduce runtime.

AUTHOR CONTRIBUTIONS

Hyrum J. Hansen: Implementation; analysis; manuscript preparation. **Stephen J. Walsh**: Conception; design; implementation; analysis; manuscript preparation. **Rong Pan**: Conception; design; manuscript preparation.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Douglas C. Montgomery and Quality and Reliability Engineering International for an excellent peer-review experience.

DATA AVAILABILITY STATEMENT

All code and data presented in this paper can be found at the author's Github: https://github.com/HyrumHansen/thesisResearch.

ORCID

Hyrum J. Hansen https://orcid.org/0009-0004-5512-8690
Stephen J. Walsh https://orcid.org/0000-0002-0505-648X
Rong Pan https://orcid.org/0000-0001-5171-8248

REFERENCES

1. Myers R, Montgomery D, Anderson-Cook C. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 4th ed. John Wiley and Sons, Ltd.; 2016.
2. Henrion D, Lasserre JB. GloptiPoly: global optimization over polynomials with Matlab and SeDuMi. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*. IEEE; 2002:747-752. doi: 10.1145/779359.779363
3. Henrion D, Lasserre JB, Lofberg J. GloptiPoly 3: moments, optimization and semidefinite programming. 2007;24:761-779. doi: 10.48550/arXiv.0709.2559
4. Borkowski J. Using a genetic algorithm to generate small exact response surface designs. *J Probab Stat*. 2003;1(1):65-88.
5. Saleh M, Pan R. A clustering-based coordinate exchange algorithm for generating G-optimal experimental designs. *J Statist Comput Simulation*. 2015;86(8):1582-1604. doi: https://doi.org/10.1080/00949655.2015.1077252
6. Hernandez LN, Nachtsheim CJ. Fast computation of exact G-optimal designs via I-optimality. *Technometrics*. 2018;60(3):297-305. doi: https://doi.org/10.1080/00401706.2017.1371080
7. Walsh SJ, Borkowski JJ. Improved G-optimal designs for small exact response surface scenarios: fast and efficient generation via particle swarm optimization. *Mathematics*. 2022;10(22):4245. doi: https://doi.org/10.3390/math10224245

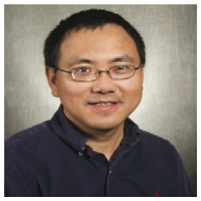
8. Goos P, Jones B. *Optimal Design of Experiments: A Case Study Approach*. Wiley; 2011. doi: https://doi.org/10.1002/9781119974017
9. Rodríguez M, Jones B, Borror CM, Montgomery DC. Generating and assessing exact G-optimal designs. *J Qual Technol*. 2010;42(1):3-20. doi: https://doi.org/10.1080/00224065.2010.11917803
10. Walsh SJ. *Development and Applications of Particle Swarm Optimization for Constructing Optimal Experimental Designs*. PhD Dissertation. Montana State UniversityBozeman, MT 59715; 2021.
11. Lasserre JB. Global optimization with polynomials and the problem of moments. *SIAM J Optim*. 2001;11(3):796-817. doi: https://doi.org/10.1137/S1052623400366802
12. Lasserre JB. Semidefinite programming vs. LP relaxations for polynomial programming. *Math Oper Res*. 2002;27(2):347-360. doi: https://doi.org/10.1287/moor.27.2.347.322
13. Lasserre JB. A semidefinite programming approach to the generalized problem of moments. *Math Program*. 2007;112(1):65-92. doi: https://doi.org/10.1007/s10107-006-0085-1
14. Kiefer J. Optimum experimental designs. *J R Stat Soc, B: Stat Methodol*. 1959;21(2):272-319. doi: https://www.jstor.org/stable/2983802
15. Kiefer J. Optimum designs in regression problems, II. *Ann Math Statist*. 1961;32(3):298-325. doi: https://www.jstor.org/stable/2237627
16. Kiefer J, Wolfowitz J. The equivalence of two extremum problems. *Can J Math*. 1960;12(4):363-366. doi: https://doi.org/10.4153/CJM-1960-030-4
17. Lasserre JB. Moments and sums of squares for polynomial optimization and related problems. *J Global Optim*. 2009;45:39-61.
18. Landau HJ. American Mathematical Society. *Moments in Mathematics*. American Mathematical Society; 1987. doi: https://doi.org/10.1090/psapm/037
19. Hernández-Lerma O, Lasserre JB. Approximation Schemes for Infinite Linear Programs. *SIAM J Optim*. 1998;8(4):973-988. doi: https://doi.org/10.1137/S1052623497315768
20. Sturm JF. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim Methods Softw*. 1999;11(1-4):625-653. doi: https://doi.org/10.1080/10556789908805766
21. Efberg J. YALMIP: a toolbox for modeling and optimization in MATLAB. In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE; 2004:284-289. doi: https://doi.org/10.1109/CACSD.2004.1393890
22. Hansen HJ. *Assessing Extant Methods for Generating G-Optimal Designs and a Novel Methodology to Compute the G-Score of a Candidate Design*. MS Thesis. Utah State UniversityLogan, UT 84321; 2024. https://digitalcommons.usu.edu/etd2023/174/
23. Meyer RK, Nachtsheim CJ. The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*. 1995;37(1):60-69. doi: https://doi.org/10.2307/1269153
24. Box GEP, Draper NR. *Response Surfaces, Mixtures, and Ridge Analyses*. John Wiley and Sons, Ltd.; 2007.
25. Khuri AE, Mukhopadhyay S. Response surface methodology. *Wire's Comput Stat*. 2010;22(2):128-149. doi: https://doi.org/10.1002/wics.73
26. Dean A, Voss D, Dragulijić D. *Design and Analysis of Experiments*. Springer; 2017.

How to cite this article: Hansen HJ, Walsh SJ, Pan R. A non-approximate method for generating *G*-optimal RSM designs. *Qual Reliab Eng Int*. 2024;1-17. https://doi.org/10.1002/qre.3660

AUTHOR BIOGRAPHIES

Hyrum J. Hansen is a post-masters researcher at Los Alamos National Lab in Los Alamos, New Mexico. He earned bachelor's degrees in political science and statistics at Utah State University and completed his MS statistics under the instruction of Dr. Stephen J. Walsh. His academic interests include mathematical optimization, Bayesian computation, design of experiments, machine learning, and applications of statistical methodology to problems in nuclear safeguards. When Hyrum is not in the lab he may be found on the edge of civilization, probably with a bike as his steed.

Dr. Stephen J. Walsh is a faculty member in the Department of Mathematics and Statistics at Utah State University. He has over a decade of experience practicing as a quality assurance and experimental design expert in laboratory environments. From 2007–2011 he was a researcher at Pacific Northwest National Laboratory in Richland, WA. From 2011–2017 he worked as a government Statistician and Quality Manager at the International Atomic Energy Agency in Vienna, Austria. His PhD, earned in 2021 from Montana State University under Dr. John J. Borkowski, provided an adaptation of the particle swarm optimization to solving several difficult exact optimal design problems. His current research program focuses on the synergistic research bridge between the design of experiments and machine learning algorithms.



Dr. Rong Pan is professor of Industrial Engineering and Data Science in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University (ASU). He is the Program Chair of Data Science, Analytics, and Engineering (DSAE) program at ASU. His research interests include failure time data analysis, design of experiments, multivariate statistical process control, time series analysis, and computational Bayesian methods.