BHT
Berliner Hochschule für Technik

Fachbereich II
Mathematik - Physik - Chemie

01/2023

Ulrike Grömping

Implementing the stratification pattern for space-filling, with dimension by weight tables

Implementieren des Stratification Patterns für Space-Filling, mit Dimension-by-Weight-Tabellen (englischsprachig)

Reports in Mathematics, Physics and Chemistry
Berichte aus der Mathematik, Physik und Chemie

ISSN (print): 2190-3913

Reports in Mathematics, Physics and Chemistry

Berichte aus der Mathematik, Physik und Chemie

The reports are freely available via the Internet:
http://www1.bht-berlin.de/FB_II/reports/welcome.htm

01/2023, August 2023

Implementing the stratification pattern for space-filling, with dimension by weight tables

Implementieren des Stratification Patterns für Space-Filling, mit Dimension-by-Weight-Tabellen (englischsprachig)

Editorial notice / Impressum

Published by / Herausgeber:
Fachbereich II
Beuth Hochschule für Technik Berlin
Luxemburger Str. 10
D-13353 Berlin
Internet: https://www.bht-berlin.de/ii

Responsibility for the content rests with the author(s) of the reports.
Die inhaltliche Verantwortung liegt bei den Autor/inn/en der Berichte.

ISSN (print): 2190-3913

Implementing the stratification pattern for space-filling, with dimension by weight tables

Ulrike Grömping

August 10th, 2023, Berliner Hochschule für Technik

Abstract

Tian and Xu proposed a stratification pattern for assessing the stratification-related space-filling qualities of stratum orthogonal arrays (SOAs) or generalizations of these (GSOAs). The pattern is directly related to (G)SOA strength. The ideas behind the stratification pattern are explained, and its implementation is presented. As a byproduct of the implementation, dimension by weight tables provide more detailed insights than the stratification pattern alone. Tian and Xu's presentation relies on a contrast matrix that contains complex elements, unless the number of levels of the GSOA is a power of 2. The complex contrasts can be replaced by a suitable real-valued coding, which fosters an understanding of the pattern for readers who are unfamiliar with complex coding. As the calculation of stratification patterns can be computationally demanding for moderately large arrays, the implementation permits to specify upper limits for dimension and/or weight, in favor of saving resources.

1 Introduction

He and Tang (2013) introduced so-called "Strong Orthogonal Arrays" (SOAs) and proposed their use for the construction of Latin Hypercube Designs (LHDs) for computer experiments. Grömping (2023) gave an overview of SOA constructions, changing the long version of the acronym to "Stratum Orthogonal Arrays", because SOAs are actually *weak* orthogonal arrays (OAs), typically of OA strength 1 only. This paper also uses the term "Stratum Orthogonal Arrays". The general idea of SOAs is to provide arrays for computer experiments with quantitative variables, with many levels for each variable. Such arrays are required to have good space-filling properties, and the introduction of SOAs is one systematic way for guaranteeing space filling by certain stratification properties: He and Tang (2013) proposed SOAs with SOA strength t (see below) for columns with s^t levels each (s a prime or prime power). An OA with OA strength 2 would require s^{2t} equireplicated level combinations for each pair of columns in s^t levels, which is usually prohibitive. SOAs make weaker requests by considering coarsened columns obtained by grouping column levels into strata of adjacent levels: s^{t-1} strata of s adjacent levels each, s^{t-2} strata of s^2 adjacent levels each, and so forth. Balance is then considered for stratum combinations. A classical SOA of strength t for m columns at s^t levels each ensures s^t equireplicated combination strata for up to t dimensions. For example, for $s = 3$ and $t = 4$, the SOA has $3^4 = 81$ equireplicated strata in 4D ($3 \times 3 \times 3 \times 3$), 81 equireplicated strata in 3D ($9 \times 3 \times 3$, $3 \times 9 \times 3$, $3 \times 3 \times 9$), 81 equireplicated strata in 2D (9×9 or 3×27 or 27×3), and 81 levels (and thus 3^4 equireplicated strata in 1D).

Requesting SOAs of strength t to have s^t levels is restrictive; variations have been considered of strength 3 with only s^2 levels (3−, introduced by Zhou and Tang 2019), strength 2 with 2D stratification properties of strength 3 (2+, introduced by He, Cheng and Tang 2018), strength 2+, but also with s^3 levels (2*, introduced by Li, Liu and Yang 2021); these can readily be extended to other strengths (e.g., strength 4− or 3+, as considered in Tian and Xu 2022 and Grömping 2023). Tian and Xu (2022) generalized SOAs to GSOAs (G for "general"), by fully separating the strength from the power to which s is taken for obtaining the number of levels.

Tian and Xu's (2022) main contribution is the so-called space-filling pattern – called stratification pattern in this paper – that has a close relationship to the generalized word length pattern of Xu and Wu (2001). In a simulation study, Tian and Xu demonstrated that superior performance on the stratification pattern was related to superior performance in evaluating a benchmark function for space-filling designs, the 8-dimensional borehole function included in the text book by Fang, Li and Sudjianto (2006). Their proposed stratification pattern is therefore worth studying.

Both the papers by Xu and Wu (2001) and by Tian and Xu (2022) used complex coding. While complex coding is often used for obtaining theoretical results, it cannot be used in data analysis. It is nevertheless possible to carry out the relevant calculations in complex coding in most professional software products, e.g., R. However, many software users including the present author are much more accustomed to real-valued contrasts and can therefore better relate to a quality criterion if they understand how it can be obtained from a real-valued model-matrix. For the GWLP by Xu and Wu (2001), it is well-known that the results are invariant to the choice of a so-called normalized orthogonal coding: the chosen complex coding is just one possible such choice. Tian and Xu's (2022) use of the complex coding is more elaborate than that of Xu and Wu (2001), because the complex coding for a basic number of levels s is used for obtaining a coding for factors in s^ℓ levels, for an $\ell \geq 1$. This paper casts their coding as a special case of a full-factorial-based coding, which fosters the understanding of how the coding works and at the same time permits easy generalization to real-valued codings with exactly the same structural properties. The implementation is available in the R package SOAs.

This paper deep-dives the stratification pattern and its underlying coding, presents an algorithm for implementing the pattern and introduces a tool for a more detailed description of the stratification structure, the so-called dimension by weight tables. The latter arises as a byproduct in the implementation. Furthermore, the paper aims at popularizing GSOAs, whose introduction is another welcome contribution by Tian and Xu (2022). The goal is to also make these results accessible to an audience that is less theoretically-minded than that of Tian and Xu (2022).

The paper proceeds as follows: Section 2 provides notation and basic facts regarding orthogonal arrays (OAs), the base s numeral system, general tools around coding in linear models and the GWLP as a predecessor and close relative of the stratification pattern. Section 3 presents Tian and Xu's (2022) contrasts, casts them in the form of full-factorial-based contrasts, and introduces their naturally-arising real-valued counterparts. Section 4 introduces Tian and Xu's GSOAs and stratification patterns and demonstrates equivalence between different variants of full-factorial-based coding of s^ℓ-level columns for obtaining a stratification pattern. Section 5 presents the implementation of the stratification pattern via full-factorial-based contrasts in R package SOAs. Small explanatory examples are placed throughout the text. Section 6 presents further examples that illustrate in particular the benefit of separating out dimensional contributions to the stratification patterns in dimension by weight tables, as well as the relations between arrays with expanded / collapsed levels and between different representations $s_1^{\ell_1} = s_2^{\ell_2}$ of the same number of levels. The discussion gives recommendations and points to opportunities for further research. The code for the examples is available as online supplementary material.

2 Notation and basic facts

$\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions. Except in the context of complex roots of the unity, where it denotes the square root of -1, the letter i is used for indices. Matrices and vectors are denoted with bold face capital or lower case letters, respectively. $\mathbf{1}_n$ and $\mathbf{0}_n$ denote a column vector of n identical elements (1 or 0), $\mathbf{I}_n$ denotes the n-dimensional identity matrix, the superscript $^\top$ denotes the transpose of a matrix or vector, and $\otimes$ the Kronecker product. Column vectors with single digit integer elements are parsimoniously written as a string of integers, e.g. $2 \cdot \mathbf{1}_5 = 22222$. The $n \times m$ matrix $\mathbf{X}$ is written as

$$\mathbf{X} = (x_{i,j})_{i=1:n,j=1:m} = \begin{pmatrix} x_{11} & \dots & x_{1m} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nm} \end{pmatrix} = (\mathbf{x}_1, \dots, \mathbf{x}_m) = \begin{pmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(n)} \end{pmatrix}.$$

Functions and unary or binary operators for scalars are applied to vectors element by element.

2.1 Orthogonal arrays and OA strength

An OA in n runs with m columns and s_i levels in the ith column is denoted as OA$(n, m, s_1 \times s_2 \times \ldots \times s_m, t)$, where t stands for the strength of the OA (explicitly called OA strength in this paper, in order to set it apart from the different concept of (G)SOA strength): OA strength t means that any tuple of t distinct columns $i_1, \dots, i_t$ of the OA has $s_{i_1} \cdot \ldots \cdot s_{i_t}$ equireplicated level combinations. An OA in n runs is *saturated*, if the main effects for its columns use $n-1$ degrees of freedom. An OA is symmetric, if $s_1 = s_2 = \cdots = s_m = s$; symmetric OAs are denoted as OA(n, m, s, t). A symmetric OA in n runs with s a prime or prime power is *regular*, if all its columns can be obtained as linear combinations of a few linearly independent basic columns from GF$(s)^n$, e.g., the k columns of an $n \times k$ full factorial design with $n = s^k$ for which all columns have the levels $0, \dots, s-1$. It is convenient to arrange the basic columns in a systematic way. Typical arrangements are lexicographic order (slow changing first, e.g., 000111222, 012012012) or fast changing first (e.g. 012012012, 000111222). Note that this is not an exhaustive definition of regular OAs, but suffices for the purpose of this paper.

2.2 Projections, coarsening, and level expansion

Let $\mathbf{D}$ be an array with n rows and m columns. Any $n \times d$ sub-matrix of $\mathbf{D}$ is called a d-dimensional projection of $\mathbf{D}$. For brevity, dimensionality is denoted as 1D, 2D, ..., or generally as dD.

Collapsing the levels $0, \dots, s^\ell - 1$ of a column in s^ℓ levels into only s^k levels $0, \dots, s^k - 1$, $k = 1, \dots, \ell - 1$, can be achieved by applying the formula $x_{s^k} = \lfloor x_{s^\ell} / (s^{\ell-k}) \rfloor$, where x_{s^ℓ} denotes the initial levels. Such coarsening groups the initial s^ℓ levels into s^k strata of adjacent levels.

Conversely, if $n = \lambda \cdot s^\ell$, $\lambda \geq 1$, the columns of an $n \times m$ array in s^k levels ($k < \ell$) can be expanded to λs^ℓ levels by

- replacing the $\lambda \cdot s^{\ell-k}$ instances of the original level 0 with values $0, \dots, s^{\ell-k} - 1$ (λ instances each),
- replacing the $\lambda \cdot s^{\ell-k}$ instances of the original level 1 with values $s^{\ell-k}, \dots, 2s^{\ell-k} - 1$ (λ instances each),
- replacing the $\lambda \cdot s^{\ell-k}$ instances of the original level 2 with values $2s^{\ell-k}, \dots, 3s^{\ell-k} - 1$ (λ instances each),
- ...
- replacing the $\lambda \cdot s^{\ell-k}$ instances of the original level $s^k - 1$ with values $(s^k - 1)s^{\ell-k}, \dots, s^\ell - 1$ (λ instances each).

Of course, expanding levels is not restricted to a power of s: one can expand the array from s^k to $\lambda' s^k$ levels for any λ' that divides $\lambda s^{\ell-k}$. In relation to (G)SOAs, expanding to a number of levels that is not a power of s has the drawback that the resulting array is not a GSOA so that a stratification pattern cannot be obtained in the usual way; however, it is of course possible to collapse levels to a power of s for obtaining a relevant stratification pattern.

Example 1. The 18×2 duplicated full factorial in two 3-level factors with columns 000111222000111222 and 012012012012012012 (i.e., $s = 3, k = 1, \ell = 2, \lambda = 2$, $s^k = 3$ levels) can be expanded to s^2 levels, e.g., to 012345678012345678 and 036147258036741852 or to 120543786201435876 and 046257174138036258; the latter expansion, like the two following level expansions, was obtained by optimizing the ϕ_p criterion for space-filling (see Section 6) using the `MDLEs` function of R package SOAs. Choosing $\lambda' = \lambda = 2$, the array can be expanded to 6 levels instead, e.g., to 000222444111333555 and 135135135024024024. Or, choosing $\lambda' = \lambda s = 6$, it can be expanded to 18 levels, e.g., to $(4, 2, 5, 10, 11, 9, 14, 16, 12, 1, 0, 3, 8, 6, 7, 17, 13, 15)^\top$ and $(3, 6, 15, 2, 10, 17, 0, 11, 14, 1, 9, 13, 5, 8, 12, 4, 7, 16)^\top$. Coarsening the levels of the resulting arrays to three levels will return the ingoing columns in each of the above cases.

The example illustrates that expanding levels can be done in different ways; these can lead to results of diverse quality. On the contrary, the coarsening of levels is a simple and unique activity.

2.3 The base s numeral system

Notations in this section are from Tian and Xu (2022). A non-negative integer $a \in \{0, 1, \dots, s^\ell - 1\}$ can be represented by an ℓ-digit number in the base s numeral system. The ith digit can be obtained with the function f_i that is defined as $f_i(a) = \lfloor a/s^{\ell-i} \rfloor \mod s$. For example, the number 5 in the base 3 numeral system with $\ell = 4$ digits is represented as 0012, as $f_1(5) = \lfloor 5/3^3 \rfloor \mod 3 = f_2(5) = \lfloor 5/3^2 \rfloor \mod 3 = 0$, $f_3(5) = \lfloor 5/3^1 \rfloor \mod 3 = 1$ and $f_4(5) = \lfloor 5/3^0 \rfloor \mod 3 = 2$. The number in the decadic system can be recovered from the base s representation via $a = \sum_{i=1}^{\ell} f_i(a) s^{\ell-i}$.

The reverse scalar product between two numbers in the base s numeral system with ℓ digits is defined as $\langle a, b \rangle = \sum_{i=1}^{\ell} f_i(a) f_{\ell-i+1}(b) = \langle a, b \rangle$; for example, $\langle 5, 5 \rangle$ in the base 3 numeral system with $\ell = 4$ digits is $0 \cdot 2 + 0 \cdot 1 + 1 \cdot 0 + 2 \cdot 0 = 0$, whereas $\langle 5, 5 \rangle$ in the base 3 numeral system with only $\ell = 2$ digits would be $1 \cdot 2 + 2 \cdot 1 = 4$ (equivalent to 1, when considered modulo 3, as is suitable for most applications).

2.4 Model matrices

As was mentioned before, GSOAs are typically used for experimentation with quantitative variables. However, investigation of their stratification behavior relies on a coding for qualitative factors. This section introduces useful terminology around qualitative coding and model matrices.

For using a qualitative factor with s levels in a linear model, its $s - 1$ main effects degrees of freedom are coded in separate model matrix columns, e.g. by dummy coding, polynomial coding or Helmert coding. Xu and Wu (2001) used the afore-mentioned normalized orthogonal coding, which is now formally defined:

Definition 1 (Normalized orthogonal coding)**.** An $s \times (s-1)$ matrix $\mathbf{C}$ is a *normalized orthogonal contrast matrix* for a factor in s levels, if all its columns have sum zero, are pairwise uncorrelated, and have squared norm s.
A model matrix for a factorial model for n runs in m factors is in *normalized orthogonal coding*, if its first column is $\mathbf{1}_n$ (for the intercept), the main effect for each factor is coded with a normalized orthogonal contrast matrix, and the $(s_1 - 1) \cdot \ldots \cdot (s_d - 1)$ d-factor interaction columns for any dD projection are obtained as the element-wise products of all combinations of main effect columns from d distinct factors.

Example 2. This paper uses three specific instances of normalized orthogonal coding for an s-level factor: coding based on the sth root of the unity, which is used by Tian and Xu (2022), normalized orthogonal polynomial coding and normalized orthogonal Helmert coding. Table 1 shows these codings for a 4-level factor. Clearly, they all fulfill the requirement that columns are orthogonal and squared norms are 4. Note that the squared norm of a complex column is the sum of the products between each element and its complex conjugate, e.g., $1^2 + i \cdot (-i) + (-1)^2 + (-i) \cdot i = 4$ for the first column in Table 1. The squared norm of a real-valued column is simply the sum of squares of its elements, e.g. $9/5 + 1/5 + 1/5 + 9/5 = 4$ for the first column of the normalized orthogonal polynomial coding of Table 1.

Let $\mathbf{F}$ denote a full factorial design in m columns at $s_1 \times \cdots \times s_m$ levels, which has $N = \prod_{i=1}^{m} s_i$ rows. For the $N \times N$ full model matrix $\mathbf{M}(\mathbf{F})$ in normalized orthogonal coding, $\mathbf{M}(\mathbf{F})^\top \mathbf{M}(\mathbf{F}) = N\mathbf{I}_N$. The full

Table 1: Complex contrasts, normalized orthogonal polynomial contrasts and normalized orthogonal Helmert contrasts for a 4-level factor.

Complex			Polynomial			Helmert		
1	2	3	1	2	3	1	2	3
$+1$	$+1$	$+1$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$
$+i$	-1	$-i$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$
-1	$+1$	-1	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	0	$\sqrt{8/3}$	$-\sqrt{1/3}$
$-i$	-1	$+i$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	0	0	$\sqrt{3}$

model matrix in normalized orthogonal coding of an actual design $\mathbf{D}$ consists of the suitable selection from the N rows of $\mathbf{M}(\mathbf{F})$; as this matrix is instrumental in assessing the properties of $\mathbf{D}$, some notation and terminology for it is now defined.

Definition 2 (Full model matrix)**.** Let $\mathbf{D}$ denote an OA with n runs and m columns.

(i) The *full model matrix* for $\mathbf{D}$ can be written as

$$\mathbf{M} = (\mathbf{M}_0, \mathbf{M}_1, \dots, \mathbf{M}_m), \tag{1}$$

where $\mathbf{M}_d$ holds all columns for the $\binom{m}{d}$ d-factor interaction effects, $d = 0, \dots, m$ (i.e., $\mathbf{M}_0 = \mathbf{1}_n$ and $\mathbf{M}_1$ holds all columns for main effects).

(ii) Within $\mathbf{M}_d, d \geq 1$, the columns for a specific dD projection are called an *effect column group.*

Example 3. Table 2 shows the model matrix for the full factorial design $\mathbf{F}$ for two 4-level factors in normalized orthogonal polynomial coding. The first column is the intercept column (not shown in the table), columns 2 to 7 contain $\mathbf{M}_1$ (three columns for each factor, i.e., two effect column groups of columns 2 to 4 and 5 to 7, respectively), and columns 8 to 15 contain $\mathbf{M}_2$ (nine columns for the only two-factor interaction, i.e., a single effect column group). Note that columns 2 to 15 of the matrix could be used as a normalized orthogonal contrast matrix for a 16-level factor. The real-valued contrasts for factors in s^ℓ levels are based on such full factorial model matrices for ℓ factors in s levels each. For using them in place of Tian and Xu's (2022) complex-valued coding, the column order has to be suitably adjusted (see Section 3.2).

2.5 The GWLP and generalized minimum aberration

The GWLP is a tool for ranking factorial designs for qualitative factors via generalized minimum aberration (GMA). It is included here for two reasons: Tian and Xu's (2022) proposal for ranking (G)SOAs follows the same spirit as GMA, and the stratification pattern and the GWLP are closely related and consist of the same ingredients.

Once the model matrix $\mathbf{M}$ of Equation (1) is available for the OA, it is straightforward to obtain the GWLP $(A_0, A_1, \dots, A_m)$. In the notation of this paper, Xu and Wu's (2001) definition of A_d amounts to

$$A_d = \mathbf{1}_n^\top \mathbf{M}_d \mathbf{M}_d^\top \mathbf{1}_n / n^2 = \sum_{\{i_1,\dots,i_d\} \subseteq \{1,\dots,m\}} a_{\{i_1,\dots,i_d\}}, \tag{2}$$

i.e., $n^2 A_d$ is the sum of the squared column sums of matrix $\mathbf{M}_d$. A_d can also be broken down into projection specific contributions $a_{\{i_1,\dots,i_d\}}$, where $a_{\{i_1,\dots,i_d\}}$ consists of only the summands belonging to columns from the effect column group for factors $i_1, \dots, i_d$.

According to Xu and Wu (2001), the A_d (and by a simple argument also the $a_{\{i_1,\dots,i_d\}}$) are invariant to

Table 2: Full model matrix for two 4-level factors in normalized orthogonal polynomial coding; the intercept column, i.e. matrix $\mathbf{M}_0(\mathbf{F})$, was omitted for saving space.

$\mathbf{M}_1(\mathbf{F})$						$\mathbf{M}_2(\mathbf{F})$								
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$9/5$	$-\sqrt{9/5}$	$3/5$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$3/5$	$-\sqrt{1/5}$	$1/5$
$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$3/5$	$-\sqrt{1/5}$	$1/5$	$\sqrt{9/5}$	-1	$\sqrt{1/5}$	$-9/5$	$\sqrt{9/5}$	$-3/5$
$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$-3/5$	$\sqrt{1/5}$	$-1/5$	$\sqrt{9/5}$	-1	$\sqrt{1/5}$	$9/5$	$-\sqrt{9/5}$	$3/5$
$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$-9/5$	$\sqrt{9/5}$	$-3/5$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-3/5$	$\sqrt{1/5}$	$-1/5$
$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$3/5$	$\sqrt{9/5}$	$-9/5$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$1/5$	$\sqrt{1/5}$	$-3/5$
$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$1/5$	$\sqrt{1/5}$	$-3/5$	$\sqrt{1/5}$	1	$-\sqrt{9/5}$	$-3/5$	$-\sqrt{9/5}$	$9/5$
$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$-1/5$	$-\sqrt{1/5}$	$3/5$	$\sqrt{1/5}$	1	$-\sqrt{9/5}$	$3/5$	$\sqrt{9/5}$	$-9/5$
$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$-3/5$	$-\sqrt{9/5}$	$9/5$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$-1/5$	$-\sqrt{1/5}$	$3/5$
$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-3/5$	$\sqrt{9/5}$	$9/5$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$-1/5$	$\sqrt{1/5}$	$3/5$
$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$-1/5$	$\sqrt{1/5}$	$3/5$	$-\sqrt{1/5}$	1	$\sqrt{9/5}$	$3/5$	$-\sqrt{9/5}$	$-9/5$
$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$1/5$	$-\sqrt{1/5}$	$-3/5$	$-\sqrt{1/5}$	1	$\sqrt{9/5}$	$-3/5$	$\sqrt{9/5}$	$9/5$
$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$3/5$	$-\sqrt{9/5}$	$-9/5$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$1/5$	$-\sqrt{1/5}$	$-3/5$
$\sqrt{9/5}$	1	$\sqrt{1/5}$	$-\sqrt{9/5}$	1	$-\sqrt{1/5}$	$-9/5$	$-\sqrt{9/5}$	$-3/5$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$-3/5$	$-\sqrt{1/5}$	$-1/5$
$\sqrt{9/5}$	1	$\sqrt{1/5}$	$-\sqrt{1/5}$	-1	$\sqrt{9/5}$	$-3/5$	$-\sqrt{1/5}$	$-1/5$	$-\sqrt{9/5}$	-1	$-\sqrt{1/5}$	$9/5$	$\sqrt{9/5}$	$3/5$
$\sqrt{9/5}$	1	$\sqrt{1/5}$	$\sqrt{1/5}$	-1	$-\sqrt{9/5}$	$3/5$	$\sqrt{1/5}$	$1/5$	$-\sqrt{9/5}$	-1	$-\sqrt{1/5}$	$-9/5$	$-\sqrt{9/5}$	$-3/5$
$\sqrt{9/5}$	1	$\sqrt{1/5}$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$9/5$	$\sqrt{9/5}$	$3/5$	$\sqrt{9/5}$	1	$\sqrt{1/5}$	$3/5$	$\sqrt{1/5}$	$1/5$

the choice of normalized orthogonal coding for $\mathbf{M}$. It is known that $A_0 = 1$ (for the intercept column), and that the sum of all A_d, $d = 0, \dots, m$, is $s_1 \cdot \dots \cdot s_m/n$ for OAs with distinct rows. The GWLP is related to the strength of an OA as follows: $A_1 = \dots = A_t = 0$ and $A_{t+1} > 0$ is equivalent to OA strength exactly t (i.e., criteria for OA strength $t+1$ are violated). Xu and Wu proposed to rank OAs by GMA, i.e. to consider an OA $\mathbf{D}_1$ as better than an OA $\mathbf{D}_2$, if the leftmost element for which the GWLPs differ is smaller for $\mathbf{D}_1$ than for $\mathbf{D}_2$.

3 Model matrices for arrays with ordered factors in s^ℓ levels

This section presents the Tian and Xu (2022) contrasts, relates them to a full-factorial-based model matrix construction, and introduces real-valued contrasts based on that construction. It also introduces Tian and Xu's concept of weights and its use in model matrices.

3.1 Tian and Xu's contrasts

Tian and Xu (2022) introduced complex-valued contrasts for factors with s^ℓ ordered levels that support evaluation of the stratification behavior of an array, assuming that equireplication between equally-sized coarser strata is more important than equireplication between equally-sized finer strata. Their contrasts are available in the R package SOAs as `contr.TianXu`. An example of their contrasts for 16 levels with $s = 4$ and $\ell = 2$ is shown in Table 3, and a formal definition is given below. The definition uses the representation of factor levels $x = 0, \dots, s^\ell - 1$ and column numbers $u = 1, \dots, s^\ell - 1$ in the base s numeral system (see Section 2.3).

Definition 3. Let $n = s^\ell$, with $s \geq 2$ and $\ell \geq 1$ integers, and let $\xi = e^{2\pi i/s}$ denote the sth root of the unity. An $s^\ell \times (s^\ell - 1)$ matrix $\mathbf{C}$ with rows labelled by levels $x = 0, \dots, s^\ell - 1$ and columns numbered with $u = 1, \dots, s^\ell - 1$ is called Tian and Xu (2022) contrast matrix, if the element in the row for level x and column u is given as the "character" $\chi_u(x) = \xi^{\langle x,u \rangle}$ with $\langle \bullet \rangle$ the reverse scalar product as defined in Section 2.3.

Example 4. Table 3 shows the Tian and Xu coding for 16 level columns with $s = 4$ and $\ell = 2$. The

Table 3: Tian and Xu contrast matrix (characters $\chi_u(x)$) for a 16-level factor based on s=4 and ℓ=2. Row labels: x values. Column labels: u values. The topmost header shows the column weights.

	1			2											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
1	$+1$	$+1$	$+1$	$+i$	$+i$	$+i$	$+i$	-1	-1	-1	-1	$-i$	$-i$	$-i$	$-i$
2	$+1$	$+1$	$+1$	-1	-1	-1	-1	$+1$	$+1$	$+1$	$+1$	-1	-1	-1	-1
3	$+1$	$+1$	$+1$	$-i$	$-i$	$-i$	$-i$	-1	-1	-1	-1	$+i$	$+i$	$+i$	$+i$
4	$+i$	-1	$-i$	$+1$	$+i$	-1	$-i$	$+1$	$+i$	-1	$-i$	$+1$	$+i$	-1	$-i$
5	$+i$	-1	$-i$	$+i$	-1	$-i$	$+1$	-1	$-i$	$+1$	$+i$	$-i$	$+1$	$+i$	-1
6	$+i$	-1	$-i$	-1	$-i$	$+1$	$+i$	$+1$	$+i$	-1	$-i$	-1	$-i$	$+1$	$+i$
7	$+i$	-1	$-i$	$-i$	$+1$	$+i$	-1	-1	$-i$	$+1$	$+i$	$+i$	-1	$-i$	$+1$
8	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1
9	-1	$+1$	-1	$+i$	$-i$	$+i$	$-i$	-1	$+1$	-1	$+1$	$-i$	$+i$	$-i$	$+i$
10	-1	$+1$	-1	-1	$+1$	-1	$+1$	$+1$	-1	$+1$	-1	-1	$+1$	-1	$+1$
11	-1	$+1$	-1	$-i$	$+i$	$-i$	$+i$	-1	$+1$	-1	$+1$	$+i$	$-i$	$+i$	$-i$
12	$-i$	-1	$+i$	$+1$	$-i$	-1	$+i$	$+1$	$-i$	-1	$+i$	$+1$	$-i$	-1	$+i$
13	$-i$	-1	$+i$	$+i$	$+1$	$-i$	-1	-1	$+i$	$+1$	$-i$	$-i$	-1	$+i$	$+1$
14	$-i$	-1	$+i$	-1	$+i$	$+1$	$-i$	$+1$	$-i$	-1	$+i$	-1	$+i$	$+1$	$-i$
15	$-i$	-1	$+i$	$-i$	-1	$+i$	$+1$	-1	$+i$	$+1$	$-i$	$+i$	$+1$	$-i$	-1

fourth root of the unity is $\xi = i = \sqrt{-1}$. The second entry ($u = 2$) in the fourth row ($x = 3$) of the matrix is one, because $x = 3$ is 03 in the base 4 numeral system with $\ell = 2$ digits, $u = 2$ is 02 in that system, and the reverse scalar product is $\langle x, u\rangle = 0 \cdot 2 + 3 \cdot 0 = 0$, which yields $\xi^0 = 1$ for the matrix entry. Likewise, the first row of the contrast matrix, which contains the entries for level $x = 0$ (00 in the base 4 numeral system), contains ones only. The 14th entry ($u = 14$) in the 15th row ($x = 14$) is also 1, because both u and x are 32 in the base 4 numeral system, so that $\langle 14, 14\rangle = 3 \cdot 2 + 2 \cdot 3 = 12$, and $\xi^{12} = \xi^0 = 1$.

3.2 Full-factorial-based contrasts

Calculating Tian and Xu's (2022) contrasts according to the definition is simple and straightforward, if one uses a software that is capable of handling complex values. For improving the understanding of how they work, as well as for achieving generalizability to real-valued contrasts for calculating stratification patterns, it is helpful to realize that Tian and Xu's construction of a contrast matrix for s^ℓ levels corresponds to obtaining a full factorial model matrix (without the intercept column) for ℓ factors in s levels each, each coded in complex coding, and with columns arranged in a way convenient for the calculation of stratification patterns. This construction is presented in the following proposition.

Proposition 1. *Tian and Xu's (2022) contrast matrix for s^ℓ level columns is equivalent to the following construction:*

(i) Denote $s^\ell \times 1$ basic factors as $\mathbf{e}_j(\ell) = \mathbf{1}_{s^{j-1}} \otimes (0, 1, \ldots, s-1)^\top \otimes \mathbf{1}_{s^{\ell-j}}$, where $j = 1, \ldots, \ell$.

(ii) Fill columns $1, \ldots, s-1$ of the $s^\ell \times (s^\ell - 1)$contrast matrix with the main effect model matrix in the complex coding for factor $\mathbf{e}_1(\ell)$ ($\xi^{\langle x,1\rangle}$ to $\xi^{\langle x,s-1\rangle}$ for level x). If $\ell = 1$, return this matrix.

(iii) For $j = 2, \ldots, \ell$, proceed as follows:

(a) Set $\mathbf{C}_{prior}$ to denote the first $s^{j-1} - 1$ columns of the contrast matrix.

(b) For $k = 1, \ldots, s-1$, fill column ks^{j-1} with the k-th main effect column for $\mathbf{e}_j(\ell)$, and the subsequent $s^{j-1} - 1$ columns with the element wise products of that column with each column from $\mathbf{C}_{prior}$.

(iv) Return the resulting matrix.

Proof. $f_j(x)$ contains the level of $\mathbf{e}_j(\ell)$.
For $u = ks^{j-1}, j = 1, \dots, \ell, k = 1, \dots, s-1$, $\langle x, u\rangle = kf_j(x)$, i.e., $\xi^{\langle u,x\rangle}$ are the $s-1$ main effects columns for $\mathbf{e}_j(\ell)$ in complex coding.
For $j \geq 2$, the $\langle x, u\rangle$ for the $s^{j-1}-1$ columns directly following column ks^{j-1} ($k = 1, \dots, s-1$) are $kf_j(x) + \sum_{i=1}^{j-1} f_i(x)f_{\ell-i+1}(u)$; hence, the model matrix columns contain the element-wise products of column ks^{j-1} with columns $1, \dots, s^{j-1}-1$, in exactly that order. Thus, the construction of the proposition is equivalent to Tian and Xu's construction. □

Definition 4 (Weight). $\rho(u) = \lceil \log_s(u+1) \rceil$, where $\log_s$ denotes the base s logarithm, is called the weight of column u.

The first $s^j - 1$ columns have at most weight j. If a column has at most weight j, its column number u can be represented in the base s numeral system using j digits (more digits are of course possible by adding leading zeroes), and the column refers to a main effect or interaction effect among the first j basic factors of Proposition 1 (i). The basic factors are sorted from slowest to fastest level changes: $\mathbf{e}_1(\ell)$ changes levels between s strata of $s^{\ell-1}$ adjacent levels only, $\mathbf{e}_2(\ell)$ changes levels between s^2 strata of $s^{\ell-2}$ adjacent levels, and so forth. Thus, the weights are related to fineness of stratification: Columns with weight 1 consider the coarsest stratification of a factor with s^ℓ ordered levels, columns with weight 2 stratify into s^2 strata of $s^{\ell-2}$ adjacent levels each, and so forth.

The following example illustrates the construction of the proposition for Tian and Xu (2022) contrasts, and their corresponding weights.

Example 5. Table 3 codes each basic factor $\mathbf{e}_1(2)$ and $\mathbf{e}_2(2)$ ($\ell = 2$) in the complex-valued 4-level contrasts of Table 1. The 16-level contrasts are then obtained from the full factorial in the two basic factors, with the first three columns as the main effect columns for $\mathbf{e}_1(2)$. These first three columns have weight 1. When moving to weight 2 columns, these first three columns constitute $\mathbf{C}_{\text{prior}}$. Column 4 is the first column with weight 2. It contains the first main effect column for $\mathbf{e}_2(2)$. The subsequent three columns contain the element-wise products of column 4 with the columns of $\mathbf{C}_{\text{prior}}$. Column 8 ($j = 2, k = 2$) holds the second main effect column for $\mathbf{e}_2(2)$, again followed by element-wise products of that column with $\mathbf{C}_{\text{prior}}$; finally, for $j = 2, k = 3$, column 12 is filled with the third main effect column of the second factor, followed by the respective interaction columns. If ℓ were larger than 2, $\mathbf{C}_{\text{prior}}$ would now be updated to the current matrix, and the next factor would be processed.

Proposition 2. *Consider Tian and Xu's complex contrasts for an s^ℓ level column, $\ell = 1, 2, \dots$. Let u denote the column number.*

(i) $u \in \{1, \dots, s-1\}$*: Column u is a main effect column for $\mathbf{e}_1(\ell)$ in the full factorial construction (slowest changing). It has up to s^1 distinct values, with the same value assigned within each of s strata of adjacent levels.*

(ii) For $\ell \geq 2$: $s^{i-1} \leq u \leq s^i - 1$, $i = 2, \dots, \ell$: Column u of the full factorial construction is a main effect column for $\mathbf{e}_i(\ell)$ or an interaction column involving $\mathbf{e}_i(\ell)$ and basic columns $\mathbf{e}_\nu(\ell), \nu < i$, where the $\mathbf{e}_\nu(\ell)$ are slower-changing than $\mathbf{e}_i(\ell)$. It has up to s^i distinct values, with the same value assigned within each of s^i strata of adjacent levels.

(iii) The exponent of s in the maximum number of distinct values according to (i) and (ii) is exactly the weight $\rho(u) = \lceil \log_s(u+1) \rceil$.

Grömping (2022) proposed real-valued contrasts for s^ℓ levels that fulfill the properties stated in Proposition 2; as these were based on saturated orthogonal arrays according to the Rao-Hamming construction, they were restricted to prime or prime power s (R function `contr.Power` in R package SOAs). Based on the construction of Proposition 1, it is straightforward to obtain a more natural set of real-valued contrasts as a replacement of Tian and Xu's (2022) complex contrasts, if real-valued contrasts are desired: simply replacing the complex contrasts in the construction with real-valued normalized orthogonal contrasts, e.g., the normalized orthogonal contrasts of Example 2. These real-valued full-factorial-based contrasts can be obtained for any integer $s \geq 2$ and are available in R package SOAs as `contr.FFbPoly` and `contr.FFbHelmert`, respectively.

Definition 5 (Full-factorial-based contrasts)**.** Proposition 1 gives rise to general contrast definitions for factors in s^ℓ levels:

(i) Contrasts obtained by the construction of Proposition 1 with any normalized orthogonal s-level coding – be it the complex coding of the proposition or any other suitable replacement – are called *full-factorial-based contrasts.*

(ii) The contrasts of Proposition 1 by Tian and Xu (2022) are called *full-factorial-based complex contrasts.*

(iii) Full-factorial-based contrasts that use the normalized orthogonal polynomial or the normalized orthogonal Helmert s-level coding are called *full-factorial-based polynomial contrasts* or *full-factorial-based Helmert contrasts*, respectively.

Example 6. Tables 4 and 5 show full-factorial-based Helmert contrasts for 4, 8, 9 or 16 levels, while Table 3 shows the full-factorial-based complex contrasts for 16 levels.
Different from the full model matrix of Definition 2 (see e.g. Table 2 for a full factorial in two 4-level factors), the columns are systematically arranged from low to high weight, according to the construction rule of Proposition 1. The weights are 122 (4 levels), 1223333 (8 levels), 11222222 (9 levels), and 111222222222222 (16 levels).
Note that the weights of Table 2 are also in ascending order, as is always the case for the conventional model matrix with $\ell \leq 2$ factors. However, the column order in Table 2 deviates from that of full-factorial-based coding: the full-factorial-based polynomial coding for a 16-level column, based on Table 2, consists of columns 2,3,4,5,8,9,10,6,11,12,13,7,14,15,16, in that order.
The contrasts for 16-level columns in Tables 3 and 5 have been based on $16 = 4^2$ ($s = 4$, $\ell = 2$). They could have also been based on $16 = 2^4$, in analogy to 4 and 8 levels. In the latter case, each single column would consist of $-1/+1$ values (even for full-factorial-based complex contrasts, because the second root of the unity is -1). In this case, the column weights would be 122333344444444. Clearly, the weights must be considered in conjunction with the choice for s, in case of different possibilities.

Table 4: Full-factorial-based Helmert contrast matrices for 4-level, 8-level and 9-level columns. The topmost header row shows the column weights.

4-level			8-level							9-level							
1	2		1	2		3				1		2					
1	2	3	1	2	3	4	5	6	7	1	2	3	4	5	6	7	8
-1	-1	$+1$	-1	-1	$+1$	-1	$+1$	$+1$	-1	$-\sqrt{3/2}$	$-\sqrt{1/2}$	$-\sqrt{3/2}$	$3/2$	$\sqrt{3/4}$	$-\sqrt{1/2}$	$\sqrt{3/4}$	$1/2$
-1	$+1$	-1	-1	-1	$+1$	$+1$	-1	-1	$+1$	$-\sqrt{3/2}$	$-\sqrt{1/2}$	$\sqrt{3/2}$	$-3/2$	$-\sqrt{3/4}$	$-\sqrt{1/2}$	$\sqrt{3/4}$	$1/2$
$+1$	-1	-1	-1	$+1$	-1	-1	$+1$	-1	$+1$	$-\sqrt{3/2}$	$-\sqrt{1/2}$	0	0	0	$\sqrt{2}$	$-\sqrt{3}$	-1
$+1$	$+1$	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$\sqrt{3/2}$	$-\sqrt{1/2}$	$-\sqrt{3/2}$	$-3/2$	$\sqrt{3/4}$	$-\sqrt{1/2}$	$-\sqrt{3/4}$	$1/2$
			$+1$	-1	-1	-1	-1	$+1$	$+1$	$\sqrt{3/2}$	$-\sqrt{1/2}$	$\sqrt{3/2}$	$3/2$	$-\sqrt{3/4}$	$-\sqrt{1/2}$	$-\sqrt{3/4}$	$1/2$
			$+1$	-1	-1	$+1$	$+1$	-1	-1	$\sqrt{3/2}$	$-\sqrt{1/2}$	0	0	0	$\sqrt{2}$	$\sqrt{3}$	-1
			$+1$	$+1$	$+1$	-1	-1	-1	-1	0	$\sqrt{2}$	$-\sqrt{3/2}$	0	$-\sqrt{3}$	$-\sqrt{1/2}$	0	-1
			$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	0	$\sqrt{2}$	$\sqrt{3/2}$	0	$\sqrt{3}$	$-\sqrt{1/2}$	0	-1
										0	$\sqrt{2}$	0	0	0	$\sqrt{2}$	0	2

Table 5: Full-factorial-based Helmert contrast matrix for 16 levels (s=4, ℓ=2). The topmost header shows the column weights.

1			2											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$-\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	$-\sqrt{2}$	2	$\sqrt{4/3}$	$\sqrt{2/3}$	$-\sqrt{2/3}$	$\sqrt{4/3}$	$2/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	$\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$-\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	$\sqrt{2}$	-2	$-\sqrt{4/3}$	$-\sqrt{2/3}$	$-\sqrt{2/3}$	$\sqrt{4/3}$	$2/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	$\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$-\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	0	0	0	0	$\sqrt{8/3}$	$-\sqrt{16/3}$	$-4/3$	$-\sqrt{8/9}$	$-\sqrt{1/3}$	$\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$-\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	0	0	0	0	0	0	0	0	$\sqrt{3}$	$-\sqrt{6}$	$-\sqrt{2}$	-1
$\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	$-\sqrt{2}$	-2	$\sqrt{4/3}$	$\sqrt{2/3}$	$-\sqrt{2/3}$	$-\sqrt{4/3}$	$2/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	$-\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	$\sqrt{2}$	2	$-\sqrt{4/3}$	$-\sqrt{2/3}$	$-\sqrt{2/3}$	$-\sqrt{4/3}$	$2/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	$-\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	0	0	0	0	$\sqrt{8/3}$	$\sqrt{16/3}$	$-4/3$	$-\sqrt{8/9}$	$-\sqrt{1/3}$	$-\sqrt{2/3}$	$\sqrt{2/9}$	$1/3$
$\sqrt{2}$	$-\sqrt{2/3}$	$-\sqrt{1/3}$	0	0	0	0	0	0	0	0	$\sqrt{3}$	$\sqrt{6}$	$-\sqrt{2}$	-1
0	$\sqrt{8/3}$	$-\sqrt{1/3}$	$-\sqrt{2}$	0	$-\sqrt{16/3}$	$\sqrt{2/3}$	$-\sqrt{2/3}$	0	$-4/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	0	$-\sqrt{8/9}$	$1/3$
0	$\sqrt{8/3}$	$-\sqrt{1/3}$	$\sqrt{2}$	0	$\sqrt{16/3}$	$-\sqrt{2/3}$	$-\sqrt{2/3}$	0	$-4/3$	$\sqrt{2/9}$	$-\sqrt{1/3}$	0	$-\sqrt{8/9}$	$1/3$
0	$\sqrt{8/3}$	$-\sqrt{1/3}$	0	0	0	0	$\sqrt{8/3}$	0	$8/3$	$-\sqrt{8/9}$	$-\sqrt{1/3}$	0	$-\sqrt{8/9}$	$1/3$
0	$\sqrt{8/3}$	$-\sqrt{1/3}$	0	0	0	0	0	0	0	0	$\sqrt{3}$	0	$\sqrt{8}$	-1
0	0	$\sqrt{3}$	$-\sqrt{2}$	0	0	$-\sqrt{6}$	$-\sqrt{2/3}$	0	0	$-\sqrt{2}$	$-\sqrt{1/3}$	0	0	-1
0	0	$\sqrt{3}$	$\sqrt{2}$	0	0	$\sqrt{6}$	$-\sqrt{2/3}$	0	0	$-\sqrt{2}$	$-\sqrt{1/3}$	0	0	-1
0	0	$\sqrt{3}$	0	0	0	0	$\sqrt{8/3}$	0	0	$\sqrt{8}$	$-\sqrt{1/3}$	0	0	-1
0	0	$\sqrt{3}$	0	0	0	0	0	0	0	0	$\sqrt{3}$	0	0	3

3.3 Using full-factorial-based contrasts for arrays with m factors in s^ℓ levels

The single weight $\rho(u)$ for the contrast column u indicates that the respective main effect model matrix column is constant within $s^{\rho(u)}$ strata of adjacent levels. The matrix $\mathbf{M}_1$ consists of $s^\ell - 1$ columns for each factor, with weights directly inherited from the contrast weights.

A model matrix column in $\mathbf{M}_d, d > 1$, is a product of main effect columns from d different factors. Thus, it relates to crossed strata obtained from the d contributing columns $u_1, \dots, u_d$ (with u_i referring to the within-effect column position), and the number of strata to be considered in d-dimensional space is, of course, $\prod_{i=1,\dots,d} s^{\rho(u_i)} = s^{\sum_{i=1,\dots,d} \rho(u_i)}$. This makes it natural to define the weight of a vector of column numbers as the sum of individual weights:

Definition 6 (Weight for a model matrix column, restated from Tian and Xu 2022)**.** Let $\mathbf{m}$ denote a column in the sub-matrix $\mathbf{M}_d$ of the model matrix $\mathbf{M}$ from Equation (1) from full-factorial-based coding for an OA with n runs and m columns in s^ℓ levels each.
Let $\mathbf{u} = (u_1, \dots, u_d)^\top$ denote the vector of column positions within their respective main effect column group for the columns that were multiplied to obtain $\mathbf{m}$.
The weight assigned to the column $\mathbf{m}$ is $\rho(\mathbf{u}) = \sum_{i=1}^d \rho(u_i)$, with $\rho(u_i)$ according to Definition 4.

Example 7. The weights shown in Tables 4 and 5 apply to main effect column groups, i.e., the $n \times (m \cdot (s^\ell - 1))$ matrix $\mathbf{M}_1$ for an array with m factors in s^ℓ levels each consists of m column groups with $s^\ell - 1$ columns each, and the weights within each such group are as stated above.
Columns of $\mathbf{M}_2$ have at least weight 2 (two main effect columns with weight 1 each) and at most weight 2ℓ (two main effect columns with weight ℓ each). Considering full factorial-based contrasts based on two 4-level factors ($s = 4, \ell = 2$, like in Tables 3 and 5), the maximum weight for the columns of $\mathbf{M}_2$ is thus 4.

4 GSOAs and the stratification pattern

As was mentioned before, in an OA of OA strength at least two with factors in s^ℓ levels, each 2D projection would consist of equireplicated copies of all conceivable $s^{2\ell}$ distinct level combinations. (G)SOAs typically have OA strength 1 only (and need not even have that strength). For evaluating their behavior in d dimensions, one considers coarsened columns, as was explained in the introduction for classical SOAs. As was mentioned before, Tian and Xu (2022) proposed GSOAs that consider s^ℓ levels in combination with strength t, where ℓ and t need not coincide. These are now defined.

Definition 7 (SOA and GSOA)**.** An array in $n = \lambda s^\ell$ runs with m columns at s^ℓ levels each is a GSOA(n, m, s^ℓ, t), $t \leq m$, if all possible stratifications into $s^t = s^{u_1} \cdot \ldots \cdot s^{u_t}$ strata, u_i integers with $0 \leq u_i \leq \ell$ and $u_1 + \ldots + u_t = t$, are equireplicated.
If $t = \ell$, the GSOA is an SOA.

For clarity, s^ℓ is always written as such, even when using specific numbers, so that it is clear from the notation whether, e.g., 81 is considered as 3^4 or as 9^2, as this difference will be relevant for the calculation of the stratification pattern. Note that the definition of GSOAs does not guarantee the existence of s^t strata in lower dimensions: for $\ell < t$, 1D (and perhaps even 2D) stratification is equireplicated (implied by balance in equireplication in higher dimensions), but may yield fewer than s^t strata (see also Example 13). For $\ell > t$, a GSOA with s^ℓ levels of strength t can, e.g., be obtained by expanding the levels of an SOA(n, m, s^t, t), assuming that s^ℓ divides n.

Example 8. A GSOA$(81, 4, 3^4, 3)$ has been created from an SOA$(81, 4, 3^3, 3)$ by expanding the levels of each column to 81 (assigning the three possible choices for each original level in random order). The ingoing SOA was obtained by the Li et al. (2021) construction from the OA$(27, 4, 3, 3)$ that is available in R package DoE.base (Grömping 2018) as `L27.3.4`. This GSOA will be revisited in Examples 9 and 10 for illustrating the stratification pattern and its interpretation.

4.1 Stratification pattern and dimension by weight table

Tian and Xu (2022) introduced the stratification pattern (the space-filling pattern of their article) as a tool to assess a (G)SOAs strength and stratification behavior, in the same spirit as the GWLP assesses an OAs strength and balance properties. The ingredients for obtaining the stratification pattern are normalized squared norms of the model matrix columns of Equation (1). For the GWLP of Xu and Wu (2001), these were summed separately for each dimension of the projection, i.e., there is a sum for each d, $d = 0, 1, \ldots, m$. For the stratification pattern, they are summed separately by the column weights instead. Of course, one can also sum them separately by combinations of dimension and weight.

Definition 8 (Stratification pattern and dimension by weight table)**.** Let $\mathbf{M}$ denote the model matrix based on full-factorial-based contrasts for an array with n runs and m columns at s^ℓ levels each. Let $\mathbf{m}$ denote a column in $\mathbf{M}_d, d = 1, \ldots, m$, and let $\mathbf{u}(\mathbf{m}) = (u_1, \ldots, u_d)^\top$ denote the vector of column positions within their main effect column group for the columns that were multiplied to obtain $\mathbf{m}$.

(i) The elements of the stratification pattern $(S_1, \ldots, S_{m\cdot\ell})$ are

$$S_j = \sum_{d=\max(1,\lceil j/\ell \rceil)}^{\min(m,j)} \sum_{\substack{\mathbf{m} \text{ a column of } \mathbf{M}_d \\ \text{and } \rho(\mathbf{u}(\mathbf{m})) = j}} (\mathbf{1}_n^\top \mathbf{m})^2 / n^2 = \sum_{d=\max(1,\lceil j/\ell \rceil)}^{\min(m,j)} s_{dj}, \tag{3}$$

where s_{dj} is short-hand for the dD inner sum.

(ii) A dimension by weight table arranges the summands s_{dj} in (3) in a table with a row for $d = 1, \ldots, m$ and a column for $j = 1, \ldots, m \cdot \ell$. Non-existent combinations (e.g., $j = 2$ with $d = 3$) are marked with ".".

The definition has been stated for any full-factorial-based coding, which requires the equivalence of all those codings. This equivalence is shown in Section 4.2.

Table 6: Dimension by weight table of contributions to GWLP and stratification pattern from the GSOA$(81, 4, 3^4, 3)$. Row labels: dimension d. Column labels: weight j. Bottom margin: stratification pattern.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	GWLP
1	0	0	0	0	.	.	.	.	.	.	.	.	.	.	.	.	0
2	.	0	0	20	32	80	96	252	.	.	.	.	.	.	.	.	480
3	.	.	0	0	8	112	368	1248	2808	5184	7776	7776	.	.	.	.	25280
4	.	.	.	0	8	20	104	504	1872	6372	17280	40176	77760	116640	139968	104976	505680
Sum	0	0	0	20	48	212	568	2004	4680	11556	25056	47952	77760	116640	139968	104976	531440

Example 9. Table 6 shows the stratification pattern (bottom row), split up by contributions from different dimensions, for the GSOA$(81, 4, 3^4, 3)$ of Example 8. There are four rows for 1D to 4D ($m = 4$), and there are 16 columns for weights $j = 1, \dots, m \cdot \ell$. The right margin shows the the sum of the row elements, which coincides with the GWLP (easy to verify with function `GWLP` of R package DoE.base). The boundaries for d in Definition 8 start at 1 for $j = 1, \dots, 4$, at 2 for $j = 5, \dots, 8$, and so forth, and go up to j for $j = 1, 2, 3$ and up to 4 for all larger j. The "." in the table indicate impossible combinations of d and j.

The following results from Tian and Xu (2022) are provided for convenience.

Lemma 1 (Tian and Xu)**.** *Consider a GSOA with n runs and m columns in s^ℓ levels each, and let $(S_1, \dots, S_{m \cdot \ell})$ denote its stratification pattern.*

(i) The GSOA has stratification strength t, if and only if $S_1 = \cdots = S_t = 0$.
(ii) If the GSOA has distinct rows, $\sum_{j=1}^{m \cdot \ell} S_j = s^{m\ell}/n - 1$.

The sum of the elements is (of course) the same as for the GWLP elements, except for omitting the 0th element that is normally included in the GWLP but not in the stratification pattern.

Example 10. We see from Table 6 that the GSOA has strength 3, and that strength 4 is violated because there are two-dimensional projections for which 3^4 equireplicated strata are not achieved. The stratification pattern for the underlying SOA$(81, 4, 3^3, 3)$ is (0, 0, 0, 20, 32, 140, 232, 616, 1056, 1440, 1728, 1296). S_4 of the GSOA is inherited from the underlying SOA; 2D stratification with weight 4 can of course not involve any single column with 81 levels.
The sum of all pattern entries for the GSOA is $81^4/81 - 1 = 531440$, i.e., one less than the sum of *all* GWLP elements (for the GWLP, including $A_0 = 1$ increases that sum by 1). It is much larger than the sum $27^4/81 - 1 = 6560$ for the underlying SOA; additional contributions from level expansion start for weight 5 and become larger with increasing weight. Example 14 in Section 6 deep-dives the relation between coarser (16) and finer (64) levels for a different case.

Tian and Xu (2022) proposed the use of a GMA like criterion for stratification patterns; this is now defined as "minimum stratification aberration" (MSA):

Definition 9 (Stratification aberration and MSA (Tian and Xu))**.** Let an array $\mathbf{D}$ with n runs and m columns in s^ℓ levels each have the stratification pattern $S(\mathbf{D}) = (S_1, S_2, \dots, S_{m \cdot \ell})$.

(i) Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be two arrays with n runs and m columns in s^ℓ levels each. $\mathbf{D}_1$ has better *stratification aberration* than $\mathbf{D}_2$, if the leftmost element for which $S(\mathbf{D}_1)$ and $S(\mathbf{D}_2)$ differ is smaller for $\mathbf{D}_1$ than for $\mathbf{D}_2$.
(ii) For a combination of n, m, s, ℓ, $\mathbf{D}$ has *minimum stratification aberration (MSA)* among all arrays in n runs with m columns in s^ℓ levels each, if there is no array with the same n, m, s, ℓ that has better stratification aberration.

The definition (of course) implies that arrays with larger strength are always better than arrays with smaller strength. Note that the definition restricts comparison to arrays with comparable properties (same n, m, s, ℓ). In particular, naïve comparisons between arrays with different numbers of levels s^ℓ may be misleading. Furthermore, where s and ℓ can not be uniquely derived from the number of levels, one has to state them with the reported stratification pattern; for example, $s = 9$ with $\ell = 2$ instead of $s = 3$ with $\ell = 4$ would have been possible in Example 9, and would have yielded a very different stratification pattern with $S_2 > 0$. Section 6 contains a detailed comparison between different choices for s and ℓ for a different example (Example 14, $16 = 4^2 = 2^4, 64 = 4^3 = 2^6$).

4.2 Equivalent use of general full-factorial-based contrasts

Definition 8 used general full-factorial-based contrasts, and it remains to be shown that these are indeed equivalent to full-factorial-based complex contrasts by Tian and Xu (2022). It is in fact straightforward to show that using general full-factorial-based contrasts, w.l.o.g. full-factorial-based polynomial contrasts, yields the exact same stratification pattern and dimension by weight table: While individual squared norms of columns of $\mathbf{M}_d$ may depend on the choice of a normalized orthogonal coding, it is known that the $a_{\{i_1,\ldots,i_d\}}$ of Equation (2) are invariant to that choice. Because of the structure of full-factorial-based contrasts, one can consider the s^ℓ-level columns as full factorials in ℓ s-level columns, and the $(s^\ell - 1)^d$ columns for a dD interaction can be subdivided into $(\sum_{i=1}^{\ell} s^{\ell-i})^d$ groups of $(s-1)^d$ columns for s-level effect interactions. The weights within such s-level effect interactions are constant. Thus, the contributions to dimension by weight tables and stratification patterns are invariant to the choice of normalized orthogonal coding in the full-factorial-based contrasts. This result is stated in a proposition:

Proposition 3. *The stratification pattern and the dimension by weight table are invariant to the specific choice of full-factorial-based complex contrasts in Definition 8.*

Note that there are further possibilities for suitable contrasts, e.g., the power contrasts proposed in Grömping (2022).

5 Implementation of the stratification pattern

In the R package SOAs (Grömping 2023b), the full-factorial-based complex contrasts are implemented in function `contr.TianXu`, the full-factorial-based polynomial contrasts in function `contr.FFbPoly`, and full-factorial-based Helmert contrasts in function `contr.FFbHelmert`. Function `Spattern` calculates the stratification pattern, and function `dim_wt_tab` allows to extract and display a dimension by weight table, like in Table 6, from a pattern created by function `Spattern`. The first subsection presents the algorithm used for implementing the stratification pattern, the second subsection discusses resources.

5.1 The algorithm

The main steps of the algorithm are presented in Table 7. The table keeps some mathematical symbols used in this paper, and switches to names of variables from the code (`in fixed width font`) for objects that only occur during computations. Note that the algorithm presented here is substantially more efficient than the earlier version reported in Grömping (2022).

The algorithm requires the design $\mathbf{D}$ and the base s as inputs. As calculations can be resource intensive for moderate or large arrays, and one is often only interested in

- low-dimensional projections (i.e., small d),
- weights j that are not much larger than t or ℓ,

the user can specify upper limits for the dimension ($d_{\max} \le m$) and the weight ($j_{\max} \le m\ell$). Per default, $j_{\max} = 4$, and $d_{\max} = j_{\max}$. If a stratification pattern has been calculated with an upper limit $d_{\max} < m$

Table 7: Algorithm for calculating the stratification pattern.

	Action	From	Details
1	Initializations		
	$\mathbf{D}, s, j_{\max}, d_{\max}$	provided	compatibility of all inputs is ascertained
	$n, m \leftarrow$	$\mathbf{D}$	numbers of rows and columns
	$\ell \leftarrow$	s, $\mathbf{D}$	($\mathbf{D}$ has s^ℓ level columns)
	$\mathbf{M}_1 \leftarrow$	s, $\mathbf{D}$	$n \times m \cdot (s^\ell - 1)$ model matrix of main effects columns, coded with full-factorial-based contrasts
	`uwt` $\leftarrow$	m, s, ℓ	$m \cdot (s^\ell - 1)$ vector of within effect weights for the columns of $\mathbf{M}_1$
	`contrib_list` $\leftarrow$	$d_{\max}$, $j_{\max}$	list; initialize as $d_{\max}$ vectors of length $j_{\max}$, `NA` entries
2	Combination preparations		
	`picks` $\leftarrow$	$m, d_{\max}$	list of $d_{\max}$ matrices for 1D to $d_{\max}$D interactions; each column holds a tuple of factor IDs to be considered for an interaction
	`cs` $\leftarrow$	$m, j_{\max}, \ell$	list of $d_{\max}$ matrices for partitions of $j_{\max}$ into $d = 1, \dots, d_{\max}$ non-zero summands; summands exceeding ℓ are reduced to ℓ, and dominated partitions are removed
	for `dim_now` in 1 to $d_{\max}$:		**loop over dimensions**
	`colnums` $\leftarrow$	`cs[[dim_now]]`, s	list of lists of `dim_now` vectors of column IDs of $\mathbf{M}_1$ for the first `dim_now` factors, restricted to those with combined weight at most $j_{\max}$ (a list of vectors for each column of `cs[[dim_now]]`)
	`colnums` $\leftarrow$	`colnums`	matrix with `dim_now` columns; all `dim_now`-tuples obtained by applying `expand.grid` to each inner list, binding the resulting data frames together, removing duplicates, and coercing the result into a matrix
	`combiweights` $\leftarrow$	`colnums`, `uwt`	list of $d_{\max}$ generic weight vectors for dD interactions, $d = 1, \dots, d_{\max}$; the jth element of the weight vector for dD refers to the column of $\mathbf{M}_d$ that is obtained as the product of the $\mathbf{M}_1$ columns indexed in the jth row of the matrix `colnums`
3	Calculations		
	for `dim_now` in 1 to $d_{\max}$:		**loop over dimensions**
	`pat_dim` $\leftarrow$	$j_{\max}$	$j_{\max}$ vector of missing values for collecting contributions from `dim_now`D projections for weights 1 to $j_{\max}$
	`picks_now` $\leftarrow$	`picks`	`picks[[dim_now]]`
	`cs_now` $\leftarrow$	`cs`	`cs[[dim_now]]`
	for `j` in 1 to `ncol(picks_now)`		**loop over projections**
	`colnums` $\leftarrow$	`cs_now`, `picks_now[,j]`, s	obtained from factors in `picks_now[,j]` in the same way as the `colnums` in part two was obtained from factors 1 to `dim_now` (this implies that the same generic weights can be used)
	for `i` in 1 to nrow(`colnums`):		**loop over `dim_now`D tuples**
	`contrib` $\leftarrow$	$\mathbf{M}_1$, `colnums[i,]`	squared norm of element-wise product of $\mathbf{M}_1$ columns
	`pat_dim` $\leftarrow$	`contrib`	add `contrib` to the `combiweights[[dim_now]][i]`th element; initialize it with this value, if it is still missing
	`contrib_list[[dim_now]]` $\leftarrow$		`pat_dim`/n^2
4	Return results		
	`aus` $\leftarrow$	`contrib_list`	sum of all elements of `contrib_list` (the stratification pattern)
	`aus` $\leftarrow$	`aus`, `contrib_list`	attach dimension by weight matrix compiled from `contrib_list` as an attribute
	`aus` $\rightarrow$	return	

on dimension, it must be interpreted w.r.t. that restriction for weights $j = d_{\max} + 1, \dots, m\ell$, whereas an upper limit on the weight has no impact on pattern entries for weights up to $j_{\max}$.

For making calculations feasible in terms of run time and memory use, the most critical step is to limit the calculation of columns of $\mathbf{M}_d, d = 1, \dots, d_{\max}$ to those that are actually needed for the requested $j_{\max}$. The algorithm handles this task for a particular d by using the dth element of the list `cs` of part 2 in Table 7 (`cs_now` of part 3 in Table 7) that contains all relevant compositions of $j_{\max}$ into d nonzero summands, with maximum element reduced to ℓ and dominated elements removed, where applicable (see Example 11 for illustration).

Example 11. For $d = 2$ and $j_{\max} = 4$, $4 = 1 + 3 = 2 + 2 = 3 + 1$.

For $\ell = 3$, this yields the columns of the matrix `cs_now` as 13, 22 and 31. The columns of $\mathbf{M}_2$ to be considered for the pair of factors 1 and 2 thus consist of

- columns with weight 1 for factor 1 (i.e., the first $s - 1$ main effect columns) and weight up to 3 for factor 2 (i.e., the first $s^3 - 1$ main effects columns)
- columns with weight up to 2 for both factors (i.e., the first $s^2 - 1$ main effects columns each)
- locums with weight up to 3 for factor 1 (the first $s^3 - 1$ main effects columns) and weight 1 for factor 2 (the first $s - 1$ main effects columns).

The different bullets contain duplications of each other: the columns with weight up to 1 in both factors are part of all three bullets. The current implementation separately creates all such sets of column combinations and subsequently removes the duplicates.

For $\ell = 2$, the maximum individual weight is two, so that the threes must be reduced to twos. Thus, the initial 13, 22 and 31 become 12, 22, and 21. As 12 and 21 are dominated by 22 (they do not imply any columns that are not implied by 22), they are eliminated from the 2nd element of the list `cs` in part 2 of Table 7, which therefore consists of the single column 22.

5.2 Use of computer resources

Run time with full-factorial-based coding has been studied for a few 64 run and 125 run examples. All calculations were conducted on a single CPU on a Windows 10 machine with 32GB RAM and an i7-12700T processor with 1.40 GHz.

For an SOA$(64, 5, 4^2, 2)$, run times with unlimited weights (i.e., $j_{\max} = m\ell = 15$) were dominated by the number of contributions from $d_{\max}$-factor interactions that contribute to the stratification pattern for the highest-dimensional projection, which can be calculated as $\binom{m}{d_{\max}} \cdot 15^{d_{\max}}$ (considering $d_{\max} = 2, \dots, 5$, with run times of 0.1, 1.8, 14.6, 51.2 seconds). For a GSOA$(64, 5, 4^3, 2)$, the $\binom{m}{d}$ portion of the number of contributions remains unchanged, but the number of columns increases by the factor $(63/15)^d$, i.e., one would expect run times (with unlimited weights) to increase to approximately 2.3, 134.1, 4547.2, 66905.2 seconds, possibly with some additional increase or decrease for overhead. Actual durations were about 1.9, 121, 3857.2 seconds for $d_{\max} = 2, 3, 4$, which is slightly faster than the calculated times. Though low dimensional projections are the most important ones, choosing a suitable maximum weight $j_{\max}$ is a better way of limiting computer resources: According to Proposition 2, for a matrix of full-factorial-based contrasts for s^ℓ levels, the number of main effect model matrix columns with weight up to $j \leq \ell$ is $s^j - 1$ (out of $s^\ell - 1$), so that limiting the overall weight by $j_{\max}$ can have a strong effect on run times via reducing the number of columns to consider in part 3 of the algorithm. Furthermore, limiting the weight to $j_{\max}$ also implies a limit of $d_{\max} = j_{\max}$. Unlimited weights imply dramatically larger run times versus limited weights, e.g., more than an hour for $d_{\max} = 4$ with unlimited weights, compared to less than five minutes for $j_{\max} = 8$ for the 64-level design in 64 runs (see Table 8). Besides the larger effect on run time, limiting the weight and thereby only implicitly limiting dimension has the benefit that $S_1, \dots, S_{j_{\max}}$ are not reduced by omitting relevant projections. For relatively large desired $j_{\max}$ and interest in $d_{\max} < j_{\max}$ dimensions only, it may nevertheless be beneficial to limit dimension to a maximum lower than the implicit limit.

Table 8 reports run times for 64 run arrays and 125 run arrays under various conditions. The table shows results averaged over single runs of three different variants of full-factorial-based coding (complex, polynomial and Helmert). For the 64 run arrays, the pattern was calculated based on $s = 2$ or $s = 4$, with the entry for weight j referring to balance in s^j strata in up to j dimensions. Run times for $s = 2$ were obtained with maximum weights 8, 10, 12, 14, 16; these correspond to the same finest stratification as for the $s = 4$ case with weights 4 to 8, but higher maximum dimension. Generally, for the 64 run case, run times up to weight 8 (16 for 2-level) were acceptable, with everything except 2-level with weight 16 running in less than four minutes. For the 125 run array, run times for weights above 6 exceeded five minutes, and the run time for 125 levels with weight 8 even exceeded two hours. The agreement between the different full factorial based codings was relatively close: R^2 values in pairwise regressions of log times over all the timings of Table 8 are at least 99.97%, and 90%-confidence ellipses for the intercept/slope pair contain the (0/1) for all three pairwise regressions between log run times. This was the reason to treat the different codings as replicates and report their average; for $s = 2$, the different codings *are* in fact replicates.

The implementation shown in Table 7 creates the entire matrix $\mathbf{M}_1$ in part 1 (except where $j_{\max} < \ell$, which should be rare), but proceeds with computations of selected columns only for model matrices $\mathbf{M}_d, d > 1$ in part 3. In an initial implementation, R's `model.matrix` function was used for creating the entire model matrix up to dimension $d_{\max}$. While this is convenient for the programmer, it caused serious memory problems for various designs that can still be handled with the approach chosen now, particularly when a (G)SOA has many levels. Even the main effect's model matrix in part 1 of the algorithm is no longer calculated with the `model.matrix` function, because a simpler creation via a loop is faster and can be limited to columns relevant under a (theoretically relevant) weight restriction $j_{\max} < \ell$.

Note that it would be possible to apply the algorithm of Table 7 using other suitable codings. Limited testing with relatively small examples showed little difference in run times between the variants of full-factorial-based coding, including the complex coding, but a run time advantage over the power coding of Grömping (2022). Although there is little difference between the variants of full-factorial coding, paired t-tests of log run times between the three studied variants show a p-value of .07 for the complex versus the normalized Helmert coding; when excluding the $s = 2$ timings for which both codings are identical, the p-value goes down to slightly less than .05. There seems to be a tendency, also found in more detailed timing inspections not included in this paper, that the complex coding is slightly slower than the real-valued codings, particularly the Helmert variant. Therefore, the full-factorial-based Helmert

Table 8: Run times in seconds from 64 run arrays with 5 columns (studied with s=2 and s=4) and a 125 run array with 6 columns, averaged over three variants of full factorial coding. The row labels show the maximum weight (for s=4 or s=5) or half the maximum weight (for s=2).

	64 runs				125 runs	
	s=2		s=4		s=5	
	16 levels	64 levels	16 levels	64 levels	25 levels	125 levels
4	0.69	1.11	0.33	0.46	2.74	3.75
5	2.75	6.20	1.37	2.59	17.20	31.29
6	8.48	31.83	4.15	12.92	90.41	228.85
7	20.84	140.50	11.14	55.82	393.86	1469.79
8	40.05	547.77	24.49	216.97	1369.88	8261.88

coding has been made the standard coding used in function `Spattern` of R package SOAs. It may be possible to speed up timings further by using sparse matrices from the Matrix package (Bates, Mächler and Jagan 2022), as interaction matrices from Helmert coding contain many zeroes (see, e.g., Table 5).

Memory usage was not studied systematically. A run time example revealed that memory constraints can limit relevant calculations with the implementation of Table 7: For a GSOA$(64, 5, 4^3, 2)$, memory failure occurred with $d_{\max} \geq 5$ and unlimited weight (i.e., $j_{\max} = 5 \cdot 3 = 15$ for $s = 4$ or $j_{\max} = 5 \cdot 6 = 30$ for $s = 2$). The failure is related to too many relevant model matrix columns due to large values in `cs`, arising from a large number of levels combined with large weights. Such problems could in principle be addressed by an implementation with a lazy version of the `expand.grid` function (e.g., as provided by Evans 2020) or other forms of using iterators; however, such an approach sacrifices speed for memory and has so far not been pursued. For most applications, it is perfectly acceptable and absolutely recommended to specify a maximum weight, and most run time studies have been conducted with specified maximum weights.

6 Examples

All calculations have been done with R, and the code is available online. Besides the stratification pattern, the ϕ_p criterion (Morris and Mitchell 1995) is used for the assessment of space-filling:

$$\phi_p(\mathbf{X}) = \left(\sum_{\{i,j\} \subset \{1,\dots,n\}, i \neq j} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})^{-p} \right)^{1/p}. \tag{4}$$

For large p, the ϕ_p criterion can mimic the maximin distance criterion well, and the R package SOAs uses the criterion with $p = 50$ for optimization of space filling within constructions, via level permutations in construction matrices. Small values of ϕ_p imply better space-filling. The ϕ_p value is a good supplement to the stratification pattern, because it captures a different aspect of space-filling: comparing stratification patterns between arrays with different numbers of levels for each column, e.g., the GSOA$(81, 4, 3^4, 3)$ of Example 9 and the SOA$(81, 4, 3^3, 3)$ from which it was created, is of limited use, because the different numbers of levels per column between the arrays are not directly reflected in the pattern as an advantage for the array with more levels, but rather as a disadvantage because the sum of pattern entries is larger for the array with more levels ($(S_1, \dots, S_6) = (0, 0, 0, 20, 32, 140)$ for the SOA, as compared to the pattern of Table 6). The ϕ_p values do capture the fineness / coarseness of columns: $\phi_p = 0.175$ for the SOA and $\phi_p = 0.06$ for the GSOA, i.e. expanding the levels substantially reduces (=improves) the ϕ_p value.

Example 12 (Published patterns)**.** Sun, Wang and Xu (2019) provided four different arrays with three 25-level columns in 25 runs. Their stratification patterns have $m\ell = 3 \cdot 2$ elements and are given in Table 9, together with their ϕ_p values. The sum of pattern elements is $25^3/25 - 1 = 624$. The fourth array is the only one that achieves SOA strength 2. This example reproduces the patterns published by Tian and Xu (2022) (their Example 4 provides S_1 to S_4 for each of these arrays). The ϕ_p values for the arrays are reasonably similar.

Table 9: Stratification patterns and ϕ_p values of the four 25-level arrays of Sun et al. (2019)

	S_1	S_2	S_3	S_4	S_5	S_6	ϕ_p
D_1	0	0.64	26.08	85.92	191.36	320	0.111
D_2	0	1.84	22.48	89.52	190.16	320	0.092
D_3	0	0.96	25.12	86.88	191.04	320	0.091
D_4	0	0	28	84	192	320	0.1

Example 13 (Extreme case ℓ=1)**.** The stratification pattern for the OA$(81, 8, 3, 4)$, available in R package DoE.base as `L81.3.8`, is (0, 0, 0, 22.42, 23.51, 18.96, 9.88, 5.23). As $\ell = 1$, the stratification pattern has exactly $m = 8$ elements, and its elements for $j = 1, \dots, m$ coincide with the corresponding GWLP elements, as was already noted by Tian and Xu 2022 for OAs with $\ell = 1$. $S_1 = S_2 = S_3 = 0$ means that the stratification strength is 3, which implies equireplication for *up to* $s^3 = 27$ strata for 1D, 2D and 3D. Of course, there are only three strata in 1D and only nine strata in 2D, because more is simply not possible. This is an extreme example that underlines what a zero in the stratification pattern means: $S_j = 0$ if and only if all existing stratifications with up to s^j strata are equireplicated, and $S_j = 0$ does *not* imply that s^j strata exist in all dimensions up to j.

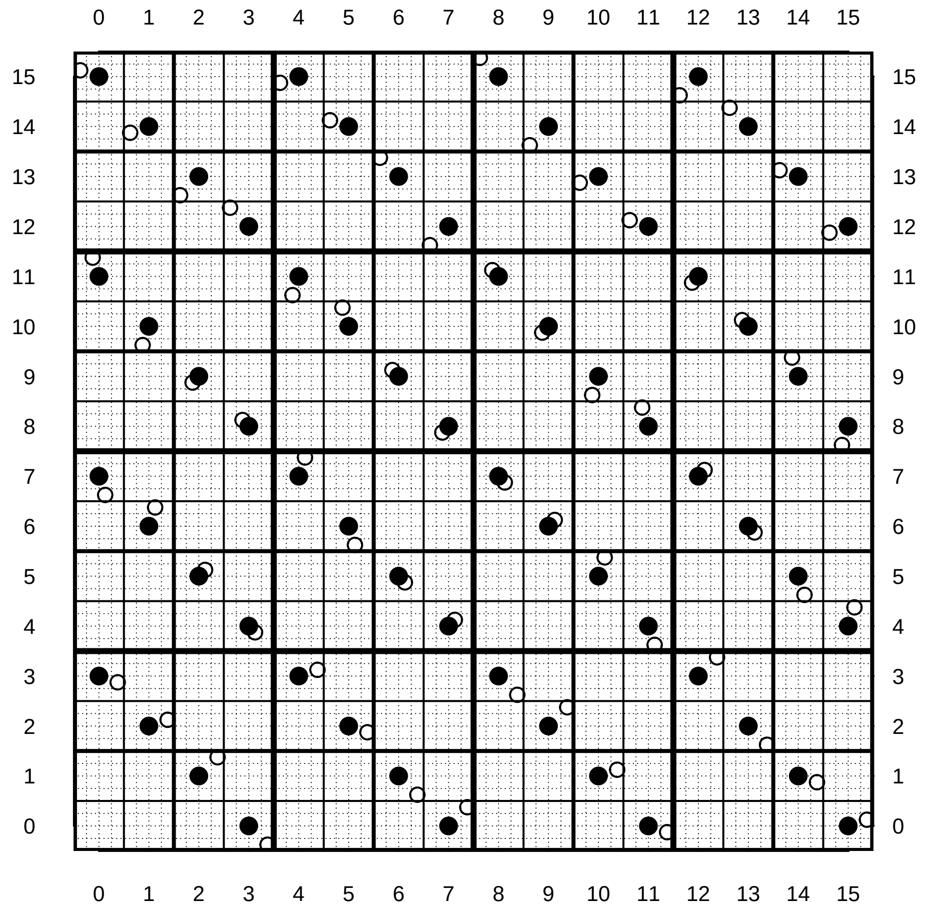

Figure 1: The first two columns of the two 64 run designs of Table 10: filled dots represent the 16-level design, circles the 64-level design. Axis labels: levels of the 16-level design; the corresponding 4 levels of the 64-level design are placed centered at these, dotted grid lines separate their positions.

We now explore the impact of collapsing or expanding levels, as well as of using larger or smaller values of s, in case different choices are possible.

Example 14 (16 versus 64 levels, and s=2 versus s=4)**.** An SOA$(64, 5, 4^3, 3)$ was constructed from a 64 run OA$(64, 6, 4, 3)$, using the construction by He and Tang (2013) (R function `SOAs`). This SOA$(64, 5, 4^3, 3)$ $(s = 4, \ell = 3)$ can also be considered as a GSOA$(64, 5, 2^6, 4)$ $(s = 2, \ell = 6)$. An array with 16 levels per column was obtained by collapsing the levels of each column; this was done for being able to compare the two arrays. Comparing arrays with different fineness / coarseness of columns is of interest, because one might choose to expand levels of an SOA for improved 1D refinement, as well as to collapse levels for saving resources when calculating a stratification pattern (see Table 8 for run times). The 16-level array is an SOA$(64, 5, 2^4, 4)$ as well as a GSOA$(64, 5, 4^2, 3)$. Thus, both arrays can be handled using $s = 4$ ($\ell = 3$ or $\ell = 2$, respectively) or using $s = 2$ ($\ell = 6$ or $\ell = 4$, respectively).

Table 10: Dimension by weight tables of contributions to stratification pattern, considering s=4 and s=2, for an SOA with 64 level columns and a version with columns collapsed to 16 levels only. Weights have been limited to 6 or 12, respectively, corresponding to considering up to 4096 strata. Row labels: dimension d. Column labels: weight j. Bottom margin: stratification pattern.

16 levels, ϕ_p=0.1000																		
	s=4						s=2											
	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	.	.	.	.	0	0	0	0	.	.	.	.	.	.	.	.
2	.	0	0	30	.	.	.	0	0	0	0	6	8	16	.	.	.	.
3	.	.	0	90	180	270	.	.	0	0	3	21	43	67	98	108	120	80
4	.	.	.	15	60	630	.	.	.	0	2	5	10	31	96	197	420	624
5	.	.	.	.	0	90	.	.	.	.	0	0	1	3	15	59	132	318
Sum	0	0	0	135	240	990	0	0	0	0	5	32	62	117	209	364	672	1022

64 levels, ϕ_p=0.0231																		
	s=4						s=2											
	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	.	.	.	0	0	0	0	0	0	.	.	.	.	.	.
2	.	0	0	90	180	360	.	0	0	0	0	18	24	68	80	120	160	160
3	.	.	0	90	360	1890	.	.	0	0	3	21	45	105	256	510	1080	2000
4	.	.	.	15	60	1050	.	.	.	0	2	5	10	35	152	361	1068	2512
5	.	.	.	.	0	90	.	.	.	.	0	0	1	3	17	69	198	628
Sum	0	0	0	195	600	3390	0	0	0	0	5	44	80	211	505	1060	2506	5300

Figure 1 shows the first two columns of the array, i.e., a 2D projection. It serves to further understanding of differences between the coarser and the finer array, regarding the stratification pattern as well as more general space-filling behavior. The filled points in the figure represent the array with 16 levels, the open points the array with 64 levels. For each array, each of the 256 squares bounded by the thin solid lines contains at most one point. For the 16 level array, 1D projections have 16 distinct points only, whereas the 1D projections for the 64 level array have 64 distinct levels. The deterioration from collapsing levels is reflected by the ϕ_p value, which is much smaller (=better) for the 64 level array ($\phi_p = 0.0231$) than for the 16 level array ($\phi_p = 0.100$). This difference in ϕ_p values exemplifies that expanding levels of an SOA to an LHS will often be desirable not so much because of stratification properties but because of other space-filling criteria. Of course, the finer levels also lead to better stratification at least in 1D, but this is not easy to detect from the stratification pattern, except for the 1D row of the dimension by weight table.

Table 10 provides a breakdown into dimensions for the first six or twelve elements of the stratification pattern for both arrays. For $s = 2$, the maximum weight has been doubled in order to consider the same maximum number of 4096 strata for both choices of s. The sum of the overall 15 or 10 ($s = 4$) or 30 or 20 ($s = 2$) elements of the stratification pattern is 16777215 for the 64-level array and 16383 for the 16 level array. Although the 64-level array does not have worse stratification behavior than the 16-level array, its stratification pattern has larger elements, which underscores that absolute entries of a stratification pattern must not be compared between arrays of different coarseness. Let us consider some entries of the dimension by weight table in more detail:

- The entries s_{dj} are the same for the finer (bottom of Table 10) and the coarser (top of Table 10) array, as long as $j \leq \ell_{\text{coarser}} + d - 1$ (i.e., $j \leq 2 + d - 1$ for $s = 4$ and $j \leq 4 + d - 1$ for $s = 2$), because the stratification possibilities for such weights j do not permit more than $s^{\ell_{\text{coarser}}}$ strata (i.e., 16 strata) for a single column even for the finer array. *For entries, where this is possible, finer levels can imply larger values:

- For $d = 2$ and $s = 2$, both arrays show perfect balance for weights up to $j = 5$, as all 2×16, 4×8, 8×4 and 16×2 stratifications exhibit perfect balance, containing 2 elements in each of the 32 strata, like in Figure 1. The entry $s_{26} = 6$ for 16 levels arises from an imbalance in 8×8 stratification for all six pairs among the first four array columns, also like in Figure 1. The s_{26} entry of the dimension by weight table for 64 levels is larger, because there are additional imbalances of 2×32 and 32×2 stratifications that cannot occur for 16 levels.

Let us now turn to comparing the $s = 4$ and $s = 2$ perspectives. With $s = 4$, both arrays have strength 3, implying that all stratifications into 4, 16 and 64 strata *by crossing columns with numbers of levels a power of* $s = 4$ (!) are fully balanced. In 2D, this means that stratification into 4×16 and 16×4 strata are fully balanced (one element each). It does *not* mean that stratification into 8×8 strata is fully balanced: we already saw in Figure 1 that this is not the case (half of the 64 8×8 strata are empty, while the other half contain two elements each). This is not in the way of $s_{23} = 0$ for $s = 4$, but the arrays cannot have $s_{26} = 0$ for $s = 2$, because weight j for $s = 4$ and weight $2j$ for $s = 2$ correspond to the same number of strata, but $s = 2$ requires to consider more types of stratifications for that number of strata: For 16 levels and $d = 2$, we have only 8×8 in addition to the variants for $s = 4$, for 64 levels we additionally have 2×32 and 32×2 that are relevant for $s = 2$ and $j = 6$ but not for $s = 4$ and $j = 3$. Hence, it is not surprising that the proportion of the total for the weights representing up to 4096 strata is larger for $s = 2$ (2483/16383=15.16% for 16 levels, 9711/16777215=0.06% for 64 levels) than for $s = 4$ (1365/16383=8.33% for 16 levels, 4185/16777215=0.02% for 64 levels).

Example 14 has illustrated how the stratification pattern

- depends on the choice of s and ℓ,
- depends on the fineness (64 levels) or coarseness (16 levels) of columns for structurally comparable arrays (related by level expansion or collapsing),
- allows refined understanding by looking at the underlying dimension by weight table,
- does not reflect all aspects of space-filling, but should be supplemented by other metrics such as ϕ_p.

7 Discussion

The GSOAs by Tian and Xu (2022) are a natural extension of SOAs. They resolve the unfortunate forced link between stratification strength and number of levels that existed in SOAs and was previously overcome by defining exceptions via qualifiers such as +, − or *. Their use for computer experiments benefits from favorable space-filling properties, and the stratification pattern (called space-filling pattern by Tian and Xu 2022) captures the stratification aspect of space-filling. The stratification pattern is a relative of the long-standing GWLP (Xu and Wu 2001): both sum the same squared sums of model matrix columns, but group the sums by different criteria (dimension for the GWLP, weight for the stratification pattern).

The stratification pattern has been implemented in the R package SOAs. Dimension by weight tabulation of the summands of the stratification pattern is a by-product of the implementation and contributes to an improved understanding of an array's stratification behavior. Due to being based on coding for qualitative factors, obtaining stratification patterns is very resource intensive for arrays with large numbers of levels; an upper limit for the weights (and thus implicitly for the projection dimensions considered) keeps computing resources in check. There may be the potential that future implementations by the group of Hongquan Xu are faster for situations that are particularly difficult for the model-matrix based approach taken in this paper; this hope is based on the relation between function `GWLP` on the one hand and function `length2` to `length5` on the other hand for obtaining GWLP elements in R package DoE.base. `GWLP` follows Hongquan Xu's approach using Krawtchouk polynomials, whereas the `length*` functions follow the model matrix approach of this paper; these functions complement each other: `GWLP` is faster for arrays that have many columns, the `length*` functions are faster for arrays that have many rows and few levels per column. As model matrices for (G)SOAs typically have very many columns, the benefit from an approach analogous to `GWLP` should be substantial.

Space-filling behavior is a multifaceted phenomenon, and there are further aspects to consider in addition to stratification behavior. For example, using the discrepancy metric ϕ_p (smaller=better, see Equation (4)) in conjunction with the stratification pattern might be a good idea. For computer experiments with quantitative variables, it can be sensible to expand the levels of an SOA to obtain a GSOA with more levels, ideally an LHD. For assessing the properties of such an expanded array, it is proposed to obtain the stratification pattern for the underlying SOA, and to assess improved space-filling of the GSOA via other criteria, e.g., the ϕ_p criterion. In this way, run time for obtaining the space-filling pattern can be kept as low as possible, without loosing relevant information.

So far, the constructions implemented in the R package SOAs permit improvements by level permutation w.r.t. the ϕ_p criterion, using an algorithm proposed by Weng (2014). The ϕ_p criterion can be cheaply calculated, and Weng's algorithm keeps searches over level permutations manageable. Improvements w.r.t. the stratification pattern would also be desirable, and level permutations have been observed to have an impact on the stratification pattern for some constructions (e.g., for the Zhou and Tang 2019 construction), but not for many others. For achieving favorable stratification behavior, improved incorporation of stratification aspects into the development of construction algorithms may be a more promising way than resource-intensive optimization via level permutation. The stratification pattern may help to instigate research in that direction.

Acknowledgments. Robert Carnell gives a home to the R package SOAs on github. Comments by anonymous reviewers prompted a re-inspection of the complex contrasts, which led to replacing the initially-chosen power contrasts (Grömping 2022) with full-factorial-based contrasts in the R-package SOAs.

References

Bates, D., Mächler, M. and Jagan, M. (2022). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.4-1. In R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Evans, B. (2020). `lazyExpandGrid.R`. Function available at https://gist.github.com/r2evans/e5531cba b8cf421d14ed.

Fang, K.-T., Li, R. and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments.* Chapman and Hall/CRC, Boca Raton, Florida.

Grömping, U. (2018). R package DoE.base for Factorial Experiments. *Journal of Statistical Software* **85**, Issue 5.

Grömping, U. (2022). Implementation of the stratification pattern by Tian and Xu via power coding. Report 3/2022, *Reports in Mathematics, Physics and Chemistry*, Department II, BHT.

Grömping, U. (2023a). A unifying implementation of stratum (aka strong) orthogonal arrays. *Computational Statistics and Data Analysis* **183**, 1-28.

Grömping, U. (2023b). SOAs: Creation of Stratum Orthogonal Arrays. R package version 1.4. In R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://github.com/bertcarnell/SOAs.

He, Y., Cheng, C.S. and Tang, B. (2018). Strong orthogonal arrays of strength two plus. *The Annals of Statistics* **46**, 457-468.

He, Y. and Tang, B. (2013). Strong orthogonal arrays and associated Latin hypercubes for computer experiments. *Biometrika* **100**, 254-260.

Hedayat, A.S., Sloane, N.J.A. and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications.* Springer-Verlag, New York. https://doi.org/10.1007/978-1-4612-1478-6.

Kuhfeld, W. (2010). Orthogonal Arrays. https://support.sas.com/techsup/technote/ts723b.pdf. Last accessed: July 04, 2022.

Li, W., Liu, M.-Q. and Yang, J.-F. (2021). Construction of column-orthogonal strong orthogonal arrays. *Statistical Papers* https://doi.org/10.1007/s00362-021-01249-w.

Liu, H. and Liu, M.-Q. (2015). Column-orthogonal strong orthogonal arrays and sliced strong orthogonal arrays. *Statistica Sinica* **25**, 1713-1734.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference* **43**, 381-402.

Owen, A. (1994). Orthogonal Arrays for Computer Experiments, Visualizations, and Integration in high dimensions. A C library available at http://lib.stat.cmu.edu/designs/oa.c.

Shi, L. and Tang, B. (2020). Construction results for strong orthogonal arrays of strength three. *Bernoulli* **26**, 418-431. https://doi.org/10.3150/19-BEJ1130

Sun, F., Wang, Y. and Xu, H. (2019). Uniform projection designs. *The Annals of Statistics* **47**, 641-661.

Tang, B. (1993). Orthogonal Array-Based Latin Hypercubes. *Journal of the American Statistical Association* **88**, 1392-1397.

Tian, Y. and Xu, H. (2022). A minimum aberration-type criterion for selecting space-filling designs. *Biometrika* **109**, 489-501.

Weng, J. (2014). Maximin Strong Orthogonal Arrays. *Master's thesis* at Simon Fraser University under supervision of Boxin Tang and Jiguo Cao.

Xu, H. and Wu, C.F.J. (2001). Generalized minimum aberration for factorial designs. *The Annals of Statistics* **29**, 1066-1077.

Zhou, Y.D. and Tang, B. (2019). Column-orthogonal strong orthogonal arrays of strength two plus and three minus. *Biometrika* **106**, 997-1004. https://doi.org/10.1093/biomet/asz043