2005-10

Very large fractional factorial and central composite designs

Sanchez, Susan M.

ACM

ACM Transactions on Modeling and Computer Sismulation (TOMACS), v.15, no.4, October

Very Large Fractional Factorial and Central Composite Designs

SUSAN M. SANCHEZ and PAUL J. SANCHEZ
Naval Postgraduate School

We present a concise representation of fractional factorials and an algorithm to quickly generate resolution V designs. The description is based on properties of a complete, orthogonal discrete-valued basis set called Walsh functions. We tabulate two-level resolution V fractional factorial designs, as well as central composite designs allowing estimation of full second-order models, for experiments involving up to 120 factors. The simple algorithm provided can be used to characterize even larger designs, and a fast Walsh transform method quickly generates design matrices from our representation.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]—*Experimental design; Statistical computing*

General Terms: Experimentation, Theory

Additional Key Words and Phrases: Design of experiments, Walsh functions, simulation experiments

1. INTRODUCTION

Factorial and fractional factorial designs have long been the mainstay of industrial experimentation. Two-level designs provide high power for testing or estimating linear effects, while fractional factorials allow this to be accomplished with fewer observations if higher-order interactions can be assumed to be negligible. Typically, only a handful of factors are dealt with in practice. Half-fractions are easy to construct, but few resolution V (R5) fractional factorial designs are readily available. For example, the largest resolution V fractional factorial in Montgomery [2000] is a 2^{10-3}, while Box et al. [1978] and NIST/SEMATECH [2005] provide a 2^{11-4}. Fractional

This work supported in part by the Marine Corps Warfighting Laboratory's Project Albert.
Authors' address: Naval Postgraduate School, 1411 Cunningham Rd., Monterey, CA 93943; email: {ssanchez, PaulSanchez}@nps.edu.

factorial designs are typically specified in terms of the so-called generators, which indicate how certain effects are confounded (aliased) with others. While it is easy to verify the confounding structure of a particular design, guidance such as "be careful in choosing the generators so that potentially important effects are not aliased with one another" [Montgomery 2000] does not indicate how appropriate generators can be easily and systematically selected for larger designs.

With advances in computing technology, experiments involving computer simulation have become more widespread. Many of the constraints on physical experiments, such as the number of factors that can be examined and controlled, need not be restrictions in the simulation setting; analysts can investigate hundreds or even thousands of factors, rather than restricting experiments to less than a dozen factors [Kleijnen et al. 2005]. Additionally, elements of the simulation model, such as the choices of distributions and their parameters, are often matters of uncertainty which may impact the system performance [Schruben et al. 1992; Chick 2001; Zouaoui and Wilson 2004; Kleijnen et al. 2005], and should be included as factors in designed experiments. Experimental designs capable of handling a large number of factors are therefore of considerable interest.

Screening experiments, in which an analyst seeks to identify a subset of the original factors that have significant main effects, are one approach. Some screening techniques can be conducted in a single stage, such as resolution III fractional factorials. Others, such as sequential bifurcation [Bettonvil and Kleijnen 1997; Wan et al. 2005] are sequential in nature; they offer potential for substantial reductions in the total sampling effort, but require assumptions such as a priori knowledge of the signs of any main effects. Screening experiments can also be used as the first stage in a series of experiments, where later stages may involve higher resolution designs. One drawback of such approaches is that factors may be excluded from further consideration that have significant impact in the form of interactions. To avoid this error, we must use designs that permit evaluation of interactions as well as main effects. However, including all possible interactions leads to explosive growth in the number of design points.

We make use of discrete-valued Walsh functions as a concise way to describe and generate factorial and fractional factorial designs. A two-level factorial or fractional factorial design can be specified using the indices of the Walsh functions. The designs can be generated easily from this representation, so large tables are unnecessary. We begin with a brief description of Walsh functions and a Walsh representation for full factorials. We then provide a simple search algorithm for generating fractional factorials, and tabulate Walsh representations for highly-fractionated resolution V fractional factorial (R5 FF) designs involving up to 120 factors in a $2^{120-105}$ design. These increase the number of factors by an order of magnitude over the standard sources [Box et al. 1978; Montgomery 2000; NIST/SEMATECH 2005], and our method readily extends to designs involving more than 120 factors. These R5 FF designs can also be used as the basis for central composite designs for experiments that investigate a very large number of factors.

2. WALSH FUNCTIONS

2.1 Overview

We provide a brief introduction to Walsh functions in this section. The interested reader can learn more from a suitable reference, such as Beauchamp [1984].

For a fixed positive integer N (a power of two) and for $i = 0, \ldots, N-1$, the ith discrete Walsh function $\mathrm{Wal}(i, n)$ is defined on the domain $\{n : n = 0, \ldots, N-1\}$ and takes the values $+1$ or -1. These can be used as column vectors of an $N \times N$ matrix. The resulting set of Walsh vectors forms a complete orthogonal basis for N-dimensional Euclidean space.

Three index sets are available, known as the sequency, dyadic, and Hadamard orderings [Beauchamp 1984]. Note that the alternative index sets are just structured reorderings; they do not yield different functions. Regardless of which ordering is used, all three obey the Walsh multiplication identity

$$\mathrm{Wal}(i, n)\mathrm{Wal}(j, n) = \mathrm{Wal}(i \oplus j, n) \qquad \text{for all } n, \tag{1}$$

where $\oplus$ denotes bitwise exclusive-OR of the integers i and j. For example, $i = 6 = 110_2$, $j = 5 = 101_2$, and $110_2 \oplus 101_2 = 011_2 = 3$, so $\mathrm{Wal}(6, n)\mathrm{Wal}(5, n) = \mathrm{Wal}(3, n)$. The $\oplus$ operator is associative. It is easy to confirm that $i \oplus i = 0$ and $i \oplus 0 = i$ for all i. From these properties it follows that $i \oplus j_1 = i \oplus j_2$ if and only if $j_1 = j_2$.

We will use the Hadamard ordering because of several nice properties. First, Hadamard ordered Walsh functions are very easy to generate in matrix form using a simple recurrence relationship. If we adopt the convention that H_m is the Hadamard ordered matrix of dimension $(2^m \times 2^m)$, then

$$H_0 = (1) \qquad \text{and} \qquad H_{m+1} = \begin{pmatrix} H_m & H_m \\ H_m & -H_m \end{pmatrix}. \tag{2}$$

Clearly, H_m is a square, symmetric matrix of dimension $2^m \times 2^m = N \times N$. For notational convenience, let $h_0, h_1, \ldots, h_{N-1}$ denote the columns of H_m. Note that the elements of h_i are $\mathrm{Wal}(i, 0), \ldots, \mathrm{Wal}(i, N-1)$.

Let Y and Θ be column vectors of dimension 2^m that constitute a transform pair with respect to H_m; that is, $Y = H_m\Theta$ and $\Theta = H_m^{-1}Y$. Then the least squares estimator for Θ is $\hat{\Theta} = (H_m' H_m)^{-1} H_m' Y$, which simplifies to $\hat{\Theta} = 2^{-m} H_m' Y$ because the columns of H_m are all mutually orthogonal and the individual matrix elements are all ± 1. Since the problem is of full rank (with no degrees of freedom for error), $\hat{\Theta} = \Theta$. Also note that since the recurrence relation is symmetric, so is H_m: $H_m' = H_m$ for all $m \geq 0$. This means that the transform pair can be represented as

$$Y = H_m\Theta; \qquad \Theta = 2^{-m} H_m Y.$$

As noted earlier, the algebraic solution is $\Theta = H_m^{-1}Y$, so $H_m^{-1} = 2^{-m}H_m$. In other words, the Hadamard-based Walsh transformation $\Theta = 2^{-m}H_m Y$ is its own inverse to within a scale constant of 2^{-m}.

There exist Fast Walsh Transforms (FWTs) similar to the more familiar Fast Fourier Transform (FFT) [Beauchamp 1984]. We will not discuss details of the

FWT algorithm, other than to note it is mathematically equivalent to multiplying the vector of inputs of dimension 2^m by $2^{-m}H_m$. We provide a Java implementation [Sanchez and Sanchez 2005] of the Hadamard ordered FWT in the Appendix.

As an example, when $N = 8$ $(= 2^3)$ we get

$$H_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix},$$

composed of column vectors $h_0, \ldots, h_7$ from left to right. It is easy to confirm that the matrix is symmetric, the columns are all mutually orthogonal and have norm 2^3, and that the row-wise product for any pair of columns obeys the Walsh multiplication identity.

2.2 Full Factorial Designs

A concise representation of factorial and fractional factorial designs is simply the total number of points N, together with a list of Walsh index assignments that match the factors to appropriate h_i's. One way (though not a unique way) to represent full factorials involving k factors is:

$$\text{assign } X_j \text{ to column } h_{i_j} \text{ where } i_j = 2^{j-1} \text{ for } j = 1, \ldots, k. \tag{3}$$

Throughout the remainder of this article, the phrase "assign Walsh index i_j to factor X_j" is used to indicate such an assignment.

The design generated by (3) follows a different ordering than the standard presentation for factorials, but there is a one-to-one correspondence [Sanchez et al. 2002]. Table I shows an analysis matrix for a 2^3 factorial and the matrix of Walsh functions with indices 1–7 when $N = 8$. Note that, in this case, the Walsh Hadamard representation is found by reversing the row index of the classical design.

The Walsh index ordering also makes it easy to determine which indices correspond to specific interactions via the multiplication identity. For example, the h_1h_2 interaction occurs at $h_{1\oplus 2} = h_3$ (Table I).

We remark that Hadamard matrices have previously appeared in the experimental design literature. For example, Chen et al. [1993] investigate non-isomorphic fractional factorials involving 16-run, 32-run, and 64-run designs; they mention that the resulting 2^{k-p} factorials can be viewed as submatrices of Hadamard matrices. Hedayat et al. [1999] have a chapter describing the mathematical properties of Hadamard functions and their relationship to Galois fields. However, the special index structure of the Hadamard-ordered Walsh functions has neither been remarked upon nor been exploited. In the next section, we present a general method for constructing and analyzing very large, highly-fractionated R5 FF designs. Our method is extremely efficient, Table I. 2^3 Factorial Design Representations

Design	Standard Representation						
	Term						
Point	X_1	X_2	X_3	X_1X_2	X_1X_3	X_2X_3	$X_1X_2X_3$
1	−	−	−	+	+	+	−
2	+	−	−	−	−	+	+
3	−	+	−	−	+	−	+
4	+	+	−	+	−	−	−
5	−	−	+	+	−	−	+
6	+	−	+	−	+	−	−
7	−	+	+	−	−	+	−
8	+	+	+	+	+	+	+

Design	Walsh Hadamard Representation						
	Index:Term						
Point	1:X_1	2:X_2	3:X_1X_2	4:X_3	5:X_1X_3	6:X_2X_3	7:$X_1X_2X_3$
1	+	+	+	+	+	+	+
2	−	+	−	+	−	+	−
3	+	−	−	+	+	−	−
4	−	−	+	+	−	−	+
5	+	+	+	−	−	−	−
6	−	+	−	−	+	−	+
7	+	−	−	−	−	+	+
8	−	−	+	−	+	+	−

easy to use, and—unlike other construction methods—allows for a concise representation of the complete information needed to generate the designs.

3. LARGE DESIGNS

3.1 Representation

The Walsh index ordering provides a concise representation of factorial or fractional factorial experiments. One need only specify N (a power of 2) and Walsh indices from among $\{j, \ldots, N-1\}$ to assign to the $k < N$ factors. Note that h_0 always estimates the mean effect. The assignments in (3) yield a full factorial design if $N = 2^k$, and a replicated full factorial if $N = 2^{k+b}$ for some $b > 0$.

3.2 Fractional Factorial Designs

Notationally, a 2^{k-p} fractional factorial means that k factors are examined, each at two levels, in a total of 2^{k-p} design points. A nearly-saturated resolution III (R3) 2^{k-p} fractional factorial can be obtained for any $k < 2^{k-p}$ as follows:

$$\text{assign Walsh index } j \text{ to factor } X_j \text{ for } j = 1, \ldots, k. \tag{4}$$

In fact, any set of k distinct indices $\{i_1, \ldots, i_k\} \in \{1, \ldots, 2^{k-p} - 1\}$ is a valid assignment; the design is saturated if $k = 2^{k-p} - 1$.

However, if the analyst wishes to be able to estimate all two-way interactions from the data, then an R5 FF or higher-resolution design is necessary. We propose a concise, recursive search algorithm, and show it can be used to identify

```
Notation
    $k$: the number of factors to be assigned
    $A$: the set of Walsh indices assigned to main effects
    $T$: the set of Walsh indices corresponding to two-way interactions occuring
        between elements of $A$

define recursive function AddFactor($a$, $k$, $A$, $T$) {
    candidate $\leftarrow$ max($i : i \in A$) + 1, where max($i : i \in \emptyset$) $\equiv 0$
    repeat {
        if ((candidate $\in T$) or (candidate $\oplus i \in A \cup T$ for some $i \in A$)) {
            candidate $\leftarrow$ candidate+1
        }
        else {
            $T \leftarrow T \cup$ {candidate $\oplus i$ for all $i \in A$}
            $A \leftarrow A \cup$ {candidate}
            if ($a < k$) {
                return AddFactor($a + 1$, $k$, $A$, $T$)
            }
            return $A$
        }
    }
}

initialize $A \leftarrow \emptyset$, $T \leftarrow \emptyset$.
input $k$
report AddFactor(1, $k$, $A$, $T$)
```

Fig. 1. Algorithm for generating 2^{k-p} resolution V fractional factorial designs.

R5 fractional factorial designs. The algorithm is provided in Figure 1, followed by a description and illustration of how it works.

The recursive function Addfactor in Algorithm 1 takes four inputs. The constants a and k indicate that the factors $X_a, \ldots, X_k$ are still awaiting index assignments, the set A contains the valid index assignments for factors X_j with $j < a$ (if any), and the set T contains the indices that correspond to the two-way interactions $X_{j_1}X_{j_2}$ such that $j_1, j_2 < a$ and $j_1 \neq j_2$. When Addfactor is invoked, it will determine a valid index assignment for the first unassigned factor (factor a) and update the sets A and T accordingly. Two conditions must hold for a candidate index assignment to be valid:

1. It cannot be an index already associated with a two-way interaction among two of the first $a - 1$ factors (candidate $\notin T$); and
2. no two-way interaction between one of the first $a-1$ factors and factor a can occur at an index already associated with either a main effect or a two-way interaction (i.e., candidate $\oplus\, i \notin A \cup T$ for all $i \in A$).

If the candidate index does not meet these two conditions, then its value is incremented by one (the next index becomes the candidate) and the process is repeated.

Once a valid index assignment is found for factor a, the set T is updated to include indices corresponding to all two-way interactions among the first a factors, and the set A is updated to include the index assignment for factor a. If $a = k$ then the algorithm terminates and the set A contains valid indices for all k factors. If $a < k$ then the Addfactor function is invoked again, with inputs indicating that factors $\{a+1, \ldots, k\}$ are still awaiting index assignments, that the (updated) set A contains indices assigned to the first a factors, and that the (updated) set T contains the indices associated with two-way interactions among the first a factors.

For illustrative purposes, we now step through the logic that Algorithm 1 uses to obtain the index assignments A for $k = 5$ factors. Algorithm 1 uses set notation for the Walsh index assignments, so any one-to-one matching of $X_1, \ldots, X_k$ to the k Walsh indices in A creates a valid design. However, for ease of exposition we use the convention that the Walsh index assignments for $X_1, X_2, \ldots, X_5$ satisfy $i_1 < i_2 < \ldots < i_5$; this corresponds to the sequence in which the indices of A are determined. No such ordering occurs as Walsh indices corresponding to the two-way interactions are added to T.

1. Initialize A and T to the empty set $\emptyset$.
2. Invoke the function Addfactor$(1, 5, \emptyset, \emptyset)$. This seeks index assignments for factors 1 through 5, beginning with factor 1. The first candidate index is $0 + 1 = 1$, and this is (trivially) a valid index assignment for factor 1. The set T remains unchanged, and the set $A \leftarrow \{1\}$.
3. Invoke the function Addfactor$(2, 5, \{1\}, \emptyset)$. This seeks index assignments for factors 2 through 5, beginning with factor 2, given that factor 1 is assigned to index 1. The first candidate index is $1 + 1 = 2$, and this is (trivially) a valid assignment for factor 2. Update T to include the two-way interaction between factors 1 and 2, which occurs at Walsh index $1 \oplus 2 = 3$. The set $A \leftarrow \{1, 2\}$.
4. Invoke the function Addfactor$(3, 5, \{1, 2\}, \{3\})$. This seeks index assignments for factors 3 through 5, beginning with factor 3, given that factors 1 and 2 are assigned to indices 1 and 2, respectively (and their interaction occurs at index 3). The first candidate index is $2 + 1 = 3$, which is not an acceptable assignment since $3 \in T$. The next candidate index is 4. Since $4 \notin T$, the main effect for factor 3 will not be confounded with the two-way interaction between factors 1 and 2. The X_1X_3 interaction would occur at $1 \oplus 4 = 5$ and the X_2X_3 interaction would occur at $2 \oplus 4 = 6$; since neither of these indices are in $A \cup T = \{1, 2, 3\}$ then this assignment is acceptable. Update $T \leftarrow \{3, 5, 6\}$ and $A \leftarrow \{1, 2, 4\}$.
5. Invoke the function Addfactor$(4, 5, \{1, 2, 4\}, \{3, 5, 6\})$. This seeks index assignments for factors 4 and 5, beginning with factor 4, given that factors 1 through 3 are assigned to indices 1, 2, and 4, respectively (and their two-way interactions occur at indices 3, 5, and 6). The candidate indices 5 and 6 are not acceptable since they are already in T. The candidate index 7 is not acceptable, since the X_1X_4 interaction would occur at $1 \oplus 7 = 6$ and this is already in T. For the candidate index 8, the X_1X_4 interaction would occur at $1 \oplus 8 = 9$, the X_2X_4 interaction would occur at $2 \oplus 8 = 10$, and the X_3X_4 interaction would occur at $4 \oplus 8 = 12$. This is acceptable, so update $T \leftarrow \{3, 5, 6, 9, 10, 12\}$ and $A \leftarrow \{1, 2, 4, 8\}$.

6. Invoke the function Addfactor(5, 5, $\{1, 2, 4, 8\}$, $\{3, 5, 6, 9, 10, 12\}$). This seeks an index assignment for factor 5, given that factors 1 through 4 are assigned to indices 1, 2, 4, and 8, respectively (and their two-way interactions occur at indices 3, 5, 6, 9, 10, 12). The candidate indices 9 and 10 are already in T so they are not acceptable. The candidate index 11 is not acceptable since the X_2X_5 interaction would occur at $2 \oplus 11 = 9$ and $9 \in T$. Index 12 is not acceptable since it is already in T, index 13 is not acceptable since the X_1X_5 interaction would occur at $1 \oplus 13 = 12$, and index 14 is not acceptable since the X_2X_5 interaction would occur at $2 \oplus 14 = 12$. For the candidate index 15, the X_1X_5, X_2X_5, X_3X_5, and X_4X_5 interactions would occur at $1 \oplus 15 = 14$, $2 \oplus 15 = 13$, $4 \oplus 15 = 11$, and $8 \oplus 15 = 7$, respectively. None of these indices is in either A or T, so this is a valid assignment. Update $T \leftarrow \{3, 5, 6, 9, 10, 12, 14, 13, 11, 7\} = \{3, 5, 6, 7, 9, 10, 11, 12, 13, 14\}$ and $A \leftarrow \{1, 2, 4, 8, 15\}$. The resulting A is the experimental design for $k = 5$ factors.

Let $A_k = \{i_1, \ldots, i_k\}$ denote the set of indices assigned to the k factors by Algorithm 1, and let $T_k = \{i \oplus j : i, j \in A_k, i \neq j\}$ denote the set of indices corresponding to two-way interactions of the k factors. As Theorem 3.2 shows, this algorithm creates R5 designs.

PROPOSITION 3.1. *If the set of index assignments A_k yields an R5 design for $N = 2^m$, then A_k also yields an R5 design for $N = 2^{m+1}$.*

PROOF. From (2), the column h_i^* of the Hadamard matrix H_{m+1} is of the form $(h_i', h_i')'$ if $i < 2^m$. Note that $i < 2^m$ for all $i \in A_k$. The columns $\{h_i^* : i \in A_k\}$ correspond to two replications of the original R5 design, so the result remains an unconfounded R5 design. □

Proposition 3.1 is used in the proof that follows.

THEOREM 3.2. *If k is a positive integer denoting the number of factors, then Algorithm 1 yields a two-level, resolution V factorial or fractional factorial design with $N = 2^{k-p}$ design points for some nonnegative integer p.*

PROOF (BY INDUCTION ON k). Assume Algorithm 1 has constructed a valid factorial or fractional factorial design for k factors (indices $\{i_1, \ldots, i_k\}$) with N_k design points, but we seek a design for $k + 1$ factors. Let i_{k+1} denote the index assigned to factor $k + 1$ by Algorithm 1, and define $i_{\max} = \max\{i : i \in A_k\}$. Algorithm 1 tests potential indices $i_{\max} + 1, i_{\max} + 2, \ldots$ until it can assign an index i_{k+1}.

Case 1: Suppose Algorithm 1 assigns an index $i_{k+1} < N_k$ to factor $k + 1$. This occurs only if (i) $i_{k+1} \notin A_k \cup T_k$, and (ii) $i_{k+1} \oplus i_j \notin A_k \cup T_k$ for all $j \leq k$. So, if $i_{k+1} < N_k$ then, by construction, A_{k+1} will also be a R5 design with N_k design points.

Case 2: Suppose that Algorithm 1 does not find a valid index $i_{k+1} < N_k$ to assign to factor $(k + 1)$. We show that a valid design exists with $N_{k+1} = 2N_k$. From Proposition 1, A_k is still a valid assignment for factors $1, \ldots, k$. The next index checked by Algorithm 1 is $i_{k+1} = N_k = 2^m$. From basic

Table II. Walsh Index Assignments for Resolution V Fractional Factorials

#	# Design						
Factors	Points	Walsh index assignments					
1	2	1					
2	4	2					
3	8	4					
4–5	16	8	15				
6	32	16					
7–8	64	32	51				
9–11	128	64	85	106			
12–17	256	128	150	171	219	237	247
18–21	512	256	279	297	455		
22–29	1,024	512	537	557	597	643	803
		863	898				
30–38	2,048	1,024	1,051	1,070	1,112	1,169	1,333
		1,345	1,620	1,866			
39–52	4,096	2,048	2,076	2,085	2,185	2,372	2,456
		2,618	2,800	2,873	3,127	3,284	3,483
		3,557	3,763				
53–69	8,192	4,096	4,125	4,135	4,174	4,435	4,459
		4,469	4,497	4,752	5,255	5,732	5,804
		5,915	6,100	6,369	6,907	7,069	
70–92	16,384	8,192	8,263	8,351	8,422	8,458	8,571
		8,750	8,858	9,124	9,314	9,500	10,026
		10,455	10,556	11,778	11,885	11,984	13,548
		14,007	14,514	14,965	15,125	15,554	
93–120	32,768	16,384	16,457	16,517	16,609	16,771	16,853
		17,022	17,453	17,891	18,073	18,562	18,980
		19,030	19,932	20,075	20,745	21,544	22,633
		23,200	24,167	25,700	26,360	26,591	26,776
		28,443	28,905	29,577	32,705		

properties of the $\oplus$ operator, and making use of the fact that $i < 2^m$ for all $i \in A_k$, we know $2^m < 2^m \oplus i < 2^{m+1}$ for all $i \in A_k$. From (1), $i_{k+1} \oplus i \notin A_k \cup T_k$ for all $i \in A_k$. This means that interactions between factor $(k+1)$ and all factors $j \leq k$ will not occur at previously assigned indices. Also, from basic properties of the $\oplus$ operator, $i_{k+1} \oplus i \neq i_{k+1} \oplus j$ for all $i \neq j$. Thus from (1), $i_{k+1} \oplus i \neq i_{k+1} \oplus j$ for every $i, j \in A_k$ with $i \neq j$. In other words, all two-way interactions involving factor $(k+1)$ and any of the first k factors occur at distinct Walsh indices greater than 2^m. It follows that $A_{k+1} \equiv A_k \cup \{2^m\}$ is an R5 design.

By inspection, if $k = 2$ then Algorithm 1 assigns indices 1 and 2 to factors 1 and 2, respectively, and returns a design with 4 design points ($N_2 = 2^2$). The X_1X_2 interaction will occur at index $1 \oplus 2 = 3$, and the result is an unconfounded full factorial design. Together with Case 1 and Case 2, this proves the result. □

Table II lists the resulting designs. To use this table, simply assign the first k indices listed in the table to the k factors. For example, if $k = 10$ a 2^{10-3} R5 FF consists of indices {1, 2, 4, 8, 15, 16, 32, 51, 64, 85}. Note that these 2^{k-p} designs have $p > 0$ for all $k \geq 5$, yielding fractional factorial designs.

The most efficient designs are those with the largest number of factors for a given number of design points (e.g., $k = 2, 3, 5, 6, 8, 11, 17, 21, 29, 38, 52, 69, 92, 120$). With this list for reference purposes, Algorithm 1 is no longer needed to identify any designs involving up to 120 factors. Note that by using all 120 indices in this table with $N = 32{,}768$, we will construct a $2^{120-105}$ R5 fractional factorial. Clearly, Algorithm 1 yields efficient (highly fractionated) designs even for very large k. It also yields a convenient method of assigning indices to factors because a single list of assignments can be used: the design for k factors (A_k) is a simple augmentation of the design for $k - 1$ factors. That is, $A_k = A_{k-1} \cup i_k$ for some index i_k.

Generating the designs from the set of indices is a straightforward task. An FWT can be called repeatedly with vectors $(0, e_i(N-1))$ for $i \in A_k$, where $e_i(N-1)$ is an elementary vector of dimension $1 \times (N-1)$ with a one in the ith location and zeroes elsewhere. The driver program for the FWT in the Appendix takes as its input the number of factors, and determines the number of design points N_k and the appropriate set of indices from Table II. It then generates the e_i columns, performs an FWT on them, and prints out the R5 FF design in a format similar to Table I, suitable for implementation and analysis. It takes only 0.2 CPU seconds on a 1.5GHz Mac PowerBook G4 to construct a 2^{12-4} R5 FF and 36 CPU seconds to construct a $2^{120-105}$ R5 FF. For comparison purposes, creating a custom design using the JMP-In software [SAS Institute 2004] on the same machine requires 60 seconds for a design involving 12 factors in $2^8 = 256$ design points, but the resulting design is not completely orthogonal; it has pairwise correlations between ± 0.0233, with the average $|\rho| = 0.00838$. The number of low levels ranged from 125–131, and only three columns had equal numbers of low and high levels. JMP took over 100 minutes to come up with an 18 factor design in 512 runs (the same size as our smallest R5 FF), but again it was not an orthogonal design: correlations ranged from -0.0158 to 0.0197, with the average $|\rho| = 0.00606$. The number of low levels ranged from 252 to 261, and only two columns had equal numbers of low and high levels. Given the dramatic increase in time required and the lack of orthogonality, we did not attempt to construct larger designs in JMP.

Once the response data (Y's) have been collected, then standard statistical regression or analysis of variance software can be used to fit a model involving all main effects and two-way interactions, and to identify which of these terms are statistically significant. An equivalent analysis option is available when the FWT is used to generate the design from the Walsh indices in Table II: taking the FWT of the response vector Y yields coefficients which, when squared and scaled by $N = 2^m$, partition the total sum of squares into components SS_i attributable to the Walsh function i ($i = 1, \ldots, N-1$). The sum squared error is

$$\mathrm{SSE} = \mathrm{SS}_{\mathrm{tot}} - \sum_{i \in A_k \cup T_k} \mathrm{SS}_i,$$

where $\mathrm{SS}_{\mathrm{tot}}$ denotes the usual total sum of squares for the given fractional factorial experiment; see, for example, p. 231 of Box et al. [1978]. The ratio $\mathrm{SS}_{i_j}/\mathrm{SSE}$ can be compared to the critical value from a t distribution with

Table III. Central Composite Design Sizes

# Factors	# Design Pts	# Factors	# Design Pts	# Factors	# Design Pts
2	10	12	284	39	4,176
3	16	17	292	52	4,202
4	26	18	550	53	8,300
5	28	21	556	69	8,332
6	46	22	1,070	70	16,526
7	80	29	1,084	92	16,570
8	82	30	2,110	93	32,956
9	150	38	2,126	120	33,010
11	154				

$N-1-\frac{k(k+1)}{2}$ degrees of freedom to test the null hypothesis of no main effect for factor j. Similarly, the ratio $\mathrm{SS}_{i_j \oplus i_h}/\mathrm{SSE}$ can be used to test for an interaction between factors j and h.

3.3 Central Composite Designs

The R5 fractional factorials of the previous section can be used as the basis for forming central composite designs as well: the 2^{k-p} R5 FF can be augmented by the addition of $2k$ star points and one or more center points. We follow Box et al. [1978] and use two center points so that all designs will have at least one degree of freedom for estimating the error variance.

Using the coded levels, the center points have the form $(X_1, \ldots, X_k) = (0, \ldots, 0)$, and the pairs of axial points have the form

$$\pm c e_j(k) \qquad \text{for } j = 1, \ldots, k$$

with values $\pm c$ at index i_j and zeroes elsewhere. Common choices for c are $c = 1$ and $c = k^{\frac{1}{2}}$. If $c = 1$, then these points occur on the faces of the k-dimensional hypercube, and the design is called a face-centered central composite; if $c = k^{\frac{1}{2}}$ this design is said to be rotatable, which means it minimizes correlation among the terms of a full second-order model [NIST/SEMATECH 2005].

Table III shows the total number of runs required for selected central composite designs—representing the smallest and largest number of factors for given N in the underlying R5 FF designs.

4. DISCUSSION

For certain values of k there are other R5 designs that require fewer runs than those in Table II. For example, Mee [2004] describes an orthogonal array that handles 47 factors with $N = 2{,}048$ runs instead of the $N = 4{,}096$ from Algorithm 1. Hedayat et al. [1999] discuss generators and error-correcting code methods that can be used to construct some R5 designs (which they call orthogonal arrays of strength 4) that involve $k = 2^m$ factors in $N = 2^{2m}$ runs. These can sometimes be more efficient than our designs when $k = 2^m$, but since our designs can use intermediate values of N for $k \neq 2^m$, neither method dominates. For example, Hedayat et al. [1999] refer to a construction that yields R5 designs for 17 to 32 factors in 1,024 runs, and designs for 33 to 64 factors in 4,096 runs. Our designs are twice as large for k in the ranges 30–32 and 53–64, but they require the same number of design points for k in the ranges 22–29 or 39–52, half as many design points for k in the ranges 18–21, 30–38, or 65–69, and one-fourth as many design points for $k = 17$.

Despite the existence of some more efficient designs for certain values of k, our method has a distinct advantage because of the ease of constructing the designs. Under 80 lines of code in the Appendix provide a method for generating R5 FFs involving up to 120 factors—where the analyst need only specify k. This code is also available online [Sanchez and Sanchez 2005]. Even without access to our implementation of the FWT, we showed in Section 1 that it is easy to generate Hadamard matrices. It is then straightforward to create a R5 FF design by selecting the set of columns specified in Table II. This is in stark contrast to the methods we have found in our literature search, where descriptions of the construction methods are complex. For example, when Hedayat et al. [1999, Section 5.11] focus on designs involving fewer than 32 factors, they describe multiple methods depending on the value of k. Mee's [2004] description of a R5 design involving 64 factors in 4,096 runs requires the analyst to (i) construct a 2^7 full factorial; (ii) for each of 32 blocks, augment the 2^7 factorial with 57 additional columns (one for each design generator, multiplied by -1 if the entry in the 57×32 blocks by foldover matrix he provides is '–'); and (iii) stack the 32 augmented blocks to obtain a 4,096-run design. While none of these steps is hard, there are many opportunities for errors to creep in if step (ii) is done manually. We see the ease of constructing very large designs for general k as a clear benefit of our approach.

Our initial interest in large designs was motivated by simulation experiments, where the time to conduct an experiment does not depend solely on the number of design points. Simulations are often characterized as having hundreds or thousands of potential factors, important interaction effects, and highly heterogeneous error variances (see, e.g., Law and Kelton [2000] and Kleijnen et al. [2005]). Once an analyst recognizes that the assumption of a common error variance may not be valid, the design must be replicated. This provides more flexibility in the choice of design. For example, suppose an analyst suspects that error variance is not constant across design points, has an interest in estimating these variances, and has sufficient time or budget for $2^{15} = 32{,}768$ runs. Partitioning this between the base design and replications yields several possibilities, ranging from 32 replications of a 2^{29-19} to 2 replications of a 2^{92-78}. If the latter is chosen and, say, 70 factors are important as main effects or in interactions, then a smaller design would not have sufficed. Conversely, if only 29 or fewer factors yield significant effects, then the analyst can eliminate 63 or more factors from further consideration and treat the data as 32 replications of a 2^{29-19}. In short, large R5 fractional factorials can be viewed as screening designs where interactions are of interest.

5. CONCLUSIONS

We have provided a concise representation for resolution V fractional factorial designs involving up to 120 factors, and a simple algorithm for generating even larger designs. This representation can be quickly converted into a design matrix for implementing an experiment.

APPENDIX

A. JAVA SOURCE CODE FOR THE HADAMARD FAST WALSH TRANSFORM

```
package walsh;

import java.text.DecimalFormat;

public class Hadamard {
   /**
    * Performs a fast Walsh transformation using a Hadamard
    * (natural) ordered in-place algorithm. Array x is altered by
    * this algorithm. If you want theresult in a new array, leaving
    * the input data intact, use the fwt(double[]) method instead.
    * <p>
    * Note that this transform is its own inverse, to within a scale
    * factor of x.length, because the transform matrix is orthogonal
    * and symmetric about its diagonal. An interesting implication
    * is that the j(th) factorial design point can be generated by
    * creating a vector of all zeros except for a one in location
    * (j-1), then performing an fwt on the vector. Repeating for
    * j=1,...,2^k yields a full factorial design for k factors
    */
   public static double[] ipfwt(double[] x) {
      double temp;
      int j;
      int k;
      int offset;
      int ngroups;

      if (!isPowerOf2(x.length)) {
         throw new IllegalArgumentException(
               "Vector length must be a power of two.");
      }
      for (int lag = 1; lag < x.length; lag = offset) {
         offset = lag << 1;
         ngroups = x.length / offset;
         for (int group = 0; group < ngroups; ++group) {
            for (int base = 0; base < lag; ++base) {
               j = base + group * offset;
               k = j + lag;
               temp = x[j];
               x[j] += x[k];
               x[k] = temp - x[k];
            }
         }
      }
      return x;
   }
```

```
/**
 * Performs a Hadamard ordered Fast Walsh Transform. This method
 * is a front-end which clones the argument array, then operates
 * on the clone in-place. Returns the resulting vector of Walsh
 * coefficients.
 */
public static double[] fwt(double[] x) {
   return ipfwt((double[]) x.clone());
}

/*
 * Private method to check whether n is a power of two
 */
private static boolean isPowerOf2(int n) {
   final int MAXINT = 1 << 30;
   for (int i = 1; i < MAXINT; i <<= 1)
      if (i == n)
         return true;
   return false;
}

/*
 * Private method to format an output string to a fixed with by
 * prepending space.
 */
private static String prependSpace(int n, String s) {
   String[] res = { "", " ", "  ", "   ", "    " };
   if (n < res.length)
      return res[n] + s;
   else
      return s;
}

/**
 * Main method to generate fractional factorial designs based
 * on pre-calculated Walsh indices from Sanchez, S. M. &
 * Sanchez, P. J., 2005, "Very large fractional factorial and
 * central composite designs," ACM Trans. Model. Comput. Simul.
 * 15(4): 362-377. User should specify the # of factors as a
 * command line argument. Program defaults to 5 factors.
 */
public static void main(String[] args) {
   String s;
   int[] index = {1, 2, 4, 8, 15, 16, 32, 51, 64, 85, 106, 128,
         150, 171, 219, 237, 247, 256, 279, 297, 455, 512, 537,
         557, 594, 643, 803, 863, 998, 1024, 1051, 1070, 1112,
         1169, 1333, 1345, 1620, 1866, 2048, 2076, 2085, 2185,
         2372, 2456, 2618, 2800, 2873, 3127, 3284, 3483, 3557,
         3763, 4096, 4125, 4135, 4174, 4435, 4459, 4469, 4497,
         4752, 5255, 5732, 5804, 5915, 6100, 6369, 6907, 7069,
         8192, 8263, 8351, 8422, 8458, 8571, 8750, 8858, 9124,
         9314, 9500, 10026, 10455, 10556, 11778, 11885, 11984,
         13548, 14007, 14514, 14965, 15125, 15554, 16384, 16457,
            16517, 16609, 16771, 16853, 17022, 17453, 17891, 18073,
            18562, 18980, 19030, 19932, 20075, 20745, 21544, 22633,
            23200, 24167, 25700, 26360, 26591, 26776, 28443, 28905,
            29577, 32705};
      int[] power = new int[index.length];
      for (int i = 0, p = 1; i < index.length; ++i) {
         if (index[i] >= p) p <<= 1;
         power[i] = p;
      }
      int size = (args.length > 0) ? Integer.parseInt(args[0]) : 5;
      double[][] design = new double[size][];
      DecimalFormat decimal0 = new DecimalFormat("###0;-##0");
      DecimalFormat decimalX = new DecimalFormat("'X'##0");
      System.out.print(" run#");
      for (int i = 1; i <= size; ++i) {
         s = decimalX.format(i);
         System.out.print(prependSpace(5 - s.length(), s));
      }
      System.out.println();
      for (int i = 0; i < size; ++i) {
         design[i] = new double[power[size-1]];
         design[i][index[i]] = 1.0;
         Hadamard.ipfwt(design[i]);
      }
      for (int j = 0; j < power[size-1]; ++j) {
         s = decimal0.format(j + 1);
         System.out.print(prependSpace(5 - s.length(), s));
         for (int i = 0; i < design.length; ++i) {
            s = decimal0.format(design[i][j]);
            System.out.print(prependSpace(5 - s.length(), s));
         }
         System.out.println();
      }
   }
}
```

REFERENCES

BEAUCHAMP, K. G. 1984. *Applications of Walsh and Related Functions, With an Introduction to Sequency Theory*. Academic Press, London.

BETTONVIL, B. J. AND KLEIJNEN, J. P. C. 1997. Searching for important factors in simulation models with many factors: Sequential bifurcation. *Eur. J. Oper. Res. 96*, 180–194.

BOX, G. E. P., HUNTER, W. G., AND HUNTER, J. S. 1978. *Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building*. Wiley, New York, NY.

CHEN, J., SUN, D. X., AND WU, C. F. J. 1993. A catalogue of two-level and three-level fractional factorial designs with small runs. *Internat. Statist. Rev. 61*, 131–145.

CHICK, S. E. 2001. Input distribution selection for simulation experiments: Accounting for input uncertainty. *Oper. Res. 49*, 5, 744–758.

HEDAYAT, A. S., SLOANE, N. J. A., AND STUFKEN, J. 1999. *Orthogonal Arrays: Theory and Applications*. Springer, New York, NY.

KLEIJNEN, J. P. C., SANCHEZ, S. M., LUCAS, T. W., AND CIOPPA, T. M. 2005. A user's guide to the brave new world of designing simulation experiments. *INFORMS J. Comput. 17*, 3, 263–289.

LAW, A. M. AND KELTON, W. D. 2000. *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, New York, NY.

MEE, R. W. 2004. Efficient two-level designs for estimating all main effects and two-factor interactions. *J. Qual. Tech. 36*, 400–412.

MONTGOMERY, D. C. 2000. *Design and Analysis of Experiments*, 5th ed. Wiley, New York, NY.

NIST/SEMATECH. 2005. *e-Handbook of Statistical Methods.* Available at <http://www.itl.nist.gov/div898/handbook/>.

SANCHEZ, P. J., HEAD, K. L., AND RAMBERG, J. S. 2002. Life in the fast lane: Yates' algorithm, fast Fourier and Walsh transforms. In *Modeling Uncertainty,* M. Dror, P. L'Ecuyer, and F. Szidarovszky, eds. Kluwer Academic Publishers, Norwell, MA, 652–684.

SANCHEZ, P. J. AND SANCHEZ, S. M. 2005. Resolution V fractional factorial generating program. Available via "Software downloads" link at <http://diana.cs.nps.navy.mil/seedlab/>. Naval Postgraduate School.

SAS INSTITUTE. 2004. *JMP-In Version 5.1 for Windows, Macintosh, and Unix*. Duxbury Thompson Learning, Pacific Grove, CA.

SCHRUBEN, L. W., SANCHEZ, S. M., SANCHEZ, P. J., AND CZITROM, V. A. 1992. Variance reallocation in Taguchi's robust design framework. In *Proceedings of the 1992 Winter Simulation Conference*, J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, Eds. IEEE, Piscataway, NJ, 548–556.

WAN, H., ANKENMAN, B., AND NELSON, B. L. 2005. Controlled sequential bifurcation: A new factor-screening method for discrete-event simulation. *Oper. Res.*, forthcoming.

ZOUAOUI, F. AND WILSON, J. R. 2004. Accounting for input-model and input-parameter uncertainties in simulation. *IIE Trans. 36*, 11, 1135–1151.

Received September 2005; revised October 2005; accepted October 2005