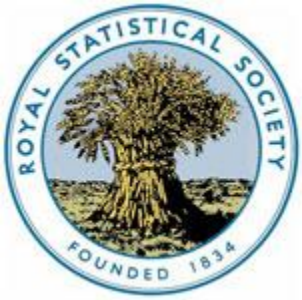

Algorithm AS 295: A Fedorov Exchange Algorithm for D-Optimal Design
Author(s): Alan J. Miller and Nam-Ky Nguyen
Source: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 43, No. 4 (1994), pp. 669-677
Published by: Wiley for the Royal Statistical Society
Stable URL: http://www.jstor.org/stable/2986264
Accessed: 23-08-2017 12:37 UTC

Appl. Statist. (1994)
43, *No.* 4, *pp.* 669–678

Algorithm AS 295

A Fedorov Exchange Algorithm for D-optimal Design

By Alan J. Miller†

CSIRO Division of Mathematics and Statistics, Clayton, Australia

and Nam-Ky Nguyen

CSIRO Biometrics Unit, Melbourne, Australia

[Received October 1990. Final revision October 1993]

Keywords: Design with blocking; D-optimality; Fedorov exchange algorithm; Optimal design

Language

Fortran 77

Description and Purpose

Optimal design algorithms are particularly useful for creating designs in difficult situations, such as when experimental conditions enforce inconvenient block sizes. Several packages are now available for computer-aided design of experiments on personal computers (PCs) either in the form of Fortran source code or as easy-to-use commercial packages. Most of these packages could not find a good design for say a $2^3 \times 3^2 \times 14$ experiment in blocks of size 10. This kind of requirement is not unusual where there are say 14 species of timber or types of cheese, and the block size is dictated by the number of experimental runs which can be carried out in one day or from one batch of material. Our algorithm allows blocking, including the use of unequal block sizes. Another important application is in augmenting an experiment which has already been carried out. Our algorithm allows some of the candidate points, those in the experiment which has been carried out, to be forced into the design.

Let X be an $N \times k$ matrix containing N possible or candidate design points. We want to find a subset of n out of the N points which maximizes the determinant

$$D = |X_n' X_n|$$

where X_n contains the corresponding n rows of X, and $n \geqslant k$. Welch (1982) produced a program for finding D-optimal designs using a branch-and-bound algorithm. This is usually only feasible for fairly small designs. His program also includes algorithms for G-optimality and for an algorithm in which the size of design makes excursions above the final design size and then returns to the desired

†*Address for correspondence*: CSIRO Division of Mathematics and Statistics, Private Bag 10, Clayton, Victoria 3169, Australia.
E-mail: alan@mel.dms.csiro.au

 0035–9254/94/43669

size. In most practical situations it is not feasible to find the global maximum of the determinants of all possible subsets of n out of N points.

As an alternative to an exhaustive search for the D-optimal design, various exchange algorithms have been proposed (see for example Fedorov (1969, 1972), Mitchell and Miller (1970), Van Schalkwyk (1971), Mitchell (1974), Galil and Kiefer (1980), Donev and Atkinson (1988) and Atkinson and Donev (1989)). Cook and Nachtsheim (1980) and Nguyen and Miller (1992) provide comparisons of these exchange algorithms, some of which are implemented in the SAS statistical package (SAS Institute, 1989). These algorithms find a local optimum, though repeated runs with different random number seeds will often find the global optimum. The Fedorov (1972) algorithm is the exchange algorithm implemented here.

Nachtsheim (1987) has reviewed commercial packages available for the computer-aided design of experiments.

Theory and Method

The usual way of computing the determinant of a matrix is by triangular factorization. The determinant of a triangular matrix is equal to the product of the elements on the diagonal. A Cholesky factorization routine, such as algorithm AS 6 (Healy, 1968), could be used to form the factorization of $X_n'X_n$. However, a more accurate result is obtained if the sum of squares and products matrix is not formed, but the Cholesky factorization is obtained directly from X_n by using an orthogonal reduction routine such as that in algorithm AS 75 (Gentleman, 1974). The routine INCLUD (algorithm AS 75.1) can be used to calculate the new Cholesky factorization each time that a point is added or removed, and this forms the basic building block of this algorithm.

To be precise, the Gentleman (1974) algorithm uses the Banachiewicz factorization

$$X_n'X_n = R'DR$$

where R is an upper triangular matrix with 1s on its diagonal, and D is a diagonal matrix. The determinant is then equal to the product of the diagonal elements of D.

As far as we are aware, these methods for calculating and updating determinants have not been used previously to find D-optimal designs. The usual methods are based around the formulae

$$|M+xx'| = |M|\,(1+x'M^{-1}x), \tag{1}$$

$$(M+xx')^{-1} = M^{-1} - \frac{M^{-1}xx'M^{-1}}{1+x'M^{-1}x} \tag{2}$$

where $M = X_n'X_n$ and x is the new point to be added to the design. If a point is to be removed from the design, all the addition and subtraction signs in equations (1) and (2) are reversed.

The disadvantages of these methods are that

(a) they cannot be used with fewer than k points in the design as M is then singular,

(b) they give poor accuracy when M is ill conditioned, e.g. when fitting polynomials in several dimensions such as quadratic surfaces, and

(c) they are also slow unless advantage is taken of the symmetry of M.

We have allowed the user to input a complete design for the algorithm to attempt to improve on. The logical variable RSTART ('random start') is set to true if the user wants the algorithm to generate a partially random starting design; otherwise the user must input the initial design in array PICKED.

The method that we use occasionally fails to find a full rank starting design. This is a very rare event and has only occurred when a full design has been input initially by the user. Restarting with a new random design has always produced a full rank design when this has been possible.

The stages of the algorithm implemented here are as follows.

(a) Unless the user is inputting a starting design, half the points in the initial design, other than any to be forced in, are chosen at random from the set of candidate points. Points are then added sequentially to maximize the rank, and to maximize the determinant for the subspace spanned by the design at that stage, to fill the design points. To speed up the process of generating a starting design, only about $\sqrt{N}$ of the candidate points are scanned for each point to be added. The same candidate point may be chosen several times; there is no attempt to force different design points.

(b) The initial design is checked to see whether it has rank k. If it is rank deficient then a point is found from the candidate points, if possible, which will increase the rank. A point which is in the current design is then found, if possible, which can be removed without reducing the rank. The two points are then exchanged. The process is repeated until either the rank is equal to k or it cannot be increased to k by using this method.

(c) Fedorov exchange algorithm: each point in this design is considered for exchange with each of the available candidate points. The pair of points chosen to exchange is the pair which maximizes the increase in the determinant. This stage is repeated until no further increase in the determinant can be obtained by a pairwise exchange.

(d) When there is more than one block, we use the interchange algorithm of Cook and Nachtsheim (1989) which uses a rank 2 update to swap selected points between blocks.

As the procedure finds a local optimum, it is usual to repeat it several times in the hope of finding the global optimum.

In stage (c), if we denote the point to be added by x_+, and that to be deleted by x_-, then the new determinant is (see for example St John and Draper (1975)):

$$|M + x_+x_+' - x_-x_-'| = |M| \{1 + \Delta(x_+, x_-)\}$$

where

$$\Delta(x_+, x_-) = x_+'M^{-1}x_+ - x_-'M^{-1}x_-(1 + x_+'M^{-1}x_+) + (x_+'M^{-1}x_-)^2. \quad (3)$$

If we write $M = R'R$, then the right-hand side can be written as

$$z_+'z_+ - z_-'z_-(1 + z_+'z_+) + (z_-'z_+)^2 \quad (4)$$

where $R'z_+ = x_+$ and $R'z_- = x_-$. The calculation of the zs from the xs is done by back-substitution. Though there are nN pairs (x_+, x_-), only the last part of the

calculation of equation (3) needs to be performed this number of times. The calculation of the vectors z_+ and z_- (and of z'_+z_+ and z'_-z_-) is only done once in each iteration for each of the N points (which include those currently in the design).

From the Cauchy–Schwarz inequality, we have that

$$(z'_-z_+)^2 \leqslant (z'_+z_+)(z'_-z_-)$$

and hence the value of expression (4) cannot be greater than $z'_+z_+ - z'_-z_-$. In searching for the best pair of points (x_+, x_-) to swap, if this quantity is smaller than the largest value of $\Delta(x_+, x_-)$ found up to that time, the full calculation of expression (4) can be skipped. The Cauchy–Schwarz inequality has been used previously by Welch (1982), but not for D-optimality.

A small time saving is achieved by not allowing the last point added to the design in one exchange to be deleted in the next exchange, and similarly by not allowing the last point deleted to be immediately reinstated.

For most experimental designs, the X-variables take only a small number of discrete values; they often have only two or three distinct values. This means that there are often many designs which yield exactly the same determinant. We found that when our DO loops for the Fedorov exchange algorithm always started at the first point in the design, or always at the first candidate point, many possible designs were never considered. In this algorithm, the DO loops start at a random point in both the selected and the candidate points. We found a substantial increase in the proportion of high determinants found for some design problems after adding this feature.

The Gentleman (1974) algorithm is particularly well suited to the Cook and Nachtsheim (1989) method for swapping elements between b blocks. We have ordered the X-variables so that the dummy (0, 1)-variables for the block factors are in the first b positions. The elements in columns $b+1$, $b+2$, . . ., k of row i are then the average values of the other variables (i.e. those which are not block factors) in block i.

On exit from routine DOPT, D and RBAR contain the orthogonal reduction corresponding to the best design found.

Structure

SUBROUTINE DOPT(X, DIM1, NCAND, KIN, N, NBLOCK, IN, BLKSIZ, K, RSTART, NRBAR, D, RBAR, PICKED, LNDET, XX, TOL, ZPZ, WK, IFAULT)

Formal parameters

X	Real array (DIM1, KIN)	input:	the array of candidate design points
DIM1	Integer	input:	the first dimension of X in the calling program; must be greater than or equal to NCAND
NCAND	Integer	input:	the number of candidate design points
KIN	Integer	input:	the number of columns in the design matrix

N	Integer	input:	the number of design points to be selected
NBLOCK	Integer	input:	the number of blocks in the design (1 for no blocks; 0 if no constant is to be fitted)
IN	Integer array (NBLOCK)	input:	array containing the number of design points to be forced into each block (often 0): set IN(1) if NBLOCK = 0
BLKSIZ	Integer array (NBLOCK)	input:	array of block sizes (if NBLOCK = 0, set BLKSIZ(1) = *N*)
K	Integer	input:	the full number of parameters to be fitted in the model, equal to KIN + NBLOCK
RSTART	Logical	input:	set to .TRUE. unless you want to enter a design
NRBAR	Integer	input:	the dimension of RBAR in the calling program; must be greater than or equal to $K(K-1)/2$
D	Real array (*K*)	output:	row multipliers in the Cholesky factorization for the selected design point
RBAR	Real array (NRBAR)	output:	the upper triangle of the Cholesky factor stored by rows omitting the diagonal elements (implicit 1s)
PICKED	Integer array (*N*)	input:	contains the initial design
		output:	contains the numbers of the chosen design points.
LNDET	Real	output:	the logarithm of the largest determinant
XX	Real array (*K*)	workspace:	holds one row of X
TOL	Real array (*K*)	workspace:	the array of tolerances
ZPZ	Real array (NCAND, NBLOCK)	workspace:	
WK	Real array (*K*)	workspace:	
IFAULT	Integer	output:	an error indicator: = −1 if no full rank starting design is found; = 0 if no error is detected; = 1* if DIM1 < NCAND; = 2* if $K < N$; = 4* if NRBAR < $K(K-1)/2$; = 8* if $K \neq$ KIN + NBLOCK; = 16* if the sum of block sizes is not equal to *N*

= 32* if any IN(I) < 0 or any IN(I) > BLKSIZ(I)

Comments

Error numbers marked with an asterisk will be summed if more than one error is detected.

If NBLOCK = 0, use dimension 1 for IN and BLKSIZ.

The user is only required to set array PICKED if an initial design is being inputted (RSTART = .FALSE.) or if points are being forced into the design, as when augmenting an existing design. If there are say three blocks of sizes 5, 10 and 10, then the first five elements of array PICKED are for the first block, the next 10 are for the second block and the third 10 are for the last block. If some points are being forced in (IN(I) > 0), then the numbers of the design points must be in the first elements of PICKED for each block. The numbers stored in array PICKED are the positions of the corresponding candidate points in array X.

If NBLOCK > 1, the second dimension of ZPZ must be at least equal to the larger of NBLOCK and 5; set the dimension equal to 1 if NBLOCK = 0.

SUBROUTINE GETX(X, DIM1, KIN, NBLOCK, K, BLOCK, XX, CASE)

Formal parameters

X, DIM1, KIN, NBLOCK, K and XX are as for subroutine DOPT.

BLOCK	Integer	input:	the number of the current block
CASE	Integer	input:	the number of the candidate point

SUBROUTINE XXTR(NP, NRBAR, D, RBAR, NREQ, TRACE, RINV)

Formal parameters

NP, NRBAR, D and RBAR are as for DOPT.

NREQ	Integer	input:	the trace of the inverse is calculated for the first NREQ rows and columns (usually NREQ = K)
TRACE	Real	output:	the required trace
RINV	Real array (NRBAR)	output:	contains the inverse of RBAR in the same format, i.e. excluding diagonal elements which are implicit 1s

Auxiliary Algorithms

A generator of uniformly distributed random numbers is required. We have used the Wichmann and Hill (1982) algorithm. If some other generator is used, the references to RAND in routine DOPT may need to be changed.

Routines CLEAR and REGCF from algorithm AS 274 (Miller, 1992) are also used.

A-optimality is sometimes used as an alternative criterion to D-optimality. This criterion is to minimize the trace of the inverse of the $X'X$ matrix. This criterion is scale dependent and hence not recommended unless the user first scales the X-variables to give an appropriate weighting to each. Routine XXTR is provided so that the user can calculate (but not minimize) this trace. It is not required to run DOPT. XXTR calls routine INV, which is not otherwise used.

The user's calling program must generate, or read from file, the array of candidate points. For block designs, dummy variables for the blocks are added in routine GETX; they should not be present in the array X supplied by the user. The user's program must also generate an initial design in the array PICKED if RSTART = .FALSE.. The ith element in this array should be set equal to the position of the corresponding element in the array of candidate points.

Constants

The real constant EPS in the DATA statement in routine DOPT is used to set up tolerances for testing for singularities. A value of 10^{-6} should be adequate in single precision for compilers which represent numbers to about seven decimal digits accuracy. A value of 10^{-15} in double precision should be adequate if numbers are represented to about 15–16 decimals.

Restrictions

The arrays X and ZPZ may be very large for some applications. When many of the columns of X are transformations of other columns, such as when interactions or polynomials are being fitted, the size of X can be substantially reduced. The subroutine GETX is called to retrieve a row from X and to place it in array XX. Routine GETX can be rewritten to calculate the interaction and polynomial terms when required, rather than to store them in X. Using this trick we have been able to use the algorithm to produce designs for a $2^{12} \times 3^2 \times 4$ experiment with specified interactions required to be estimated, on a PC.

Examples of Use

This algorithm has been tested by using several of the 'standard' problems from the optimal design literature. We have also tried several of the alternative exchange algorithms, but in our experience the Fedorov algorithm is sufficiently fast to run on a PC and gives consistently good performance compared with alternative, and often more complex, algorithms.

Firstly we consider the problem of fitting a linear model in 10 variables, with a constant, and using the minimum of 11 design points. Each variable takes only the values $+1$ or -1 at each point. The maximum possible determinant has been shown to be $25 \times 2^{32} = 107374182400$. There are $2^{10} = 1024$ candidate points to be searched. This algorithm found an optimal design (there are many equivalent permutations of the design) in 48 out of 100 tries. Running on a 16 MHz 80386-machine with an 80387 floating point processor under Lahey's F77L Fortran compiler, each try took about 1 min. As all the determinants have integer values for this problem, it was simple to test the accuracy. With double precision,

equivalent to about a 16 decimal digit representation of floating point numbers, no reported determinant was in error by more than 3 in the 15th significant digit.

As a second example, we attempted to reproduce the results in Table 3 of Galil and Kiefer (1980). This gives the maximum determinants found for fitting a quadratic surface in three, four or five variables with various numbers of design points. Table 1 shows our maximum determinants and those of Galil and Kiefer for the cases in which they were different. Each variable takes only the values -1, 0 or $+1$. We used 100 tries in each case.

The two previous examples do not include blocking. As a final example, we look at finding designs for fitting quadratic surfaces with blocking. If p is the number of variables and b is the number of blocks, then the number of parameters to be fitted is $p(p+3)/2+b$. If we allow each variable to take only the values -1, 0 or $+1$, then there are 3^p candidate points. For three variables, and four blocks each of size 8, each try took an average of 11 s on our previously mentioned PC. The largest determinant obtained was 7.228×10^{13} which occurred twice in 100 tries. There is no guarantee that the global maximum had been found, though subsequent runs have not improved on this determinant. The largest determinant from the 100 runs was only about 20% larger than the smallest. Such a difference would be unimportant in most practical applications.

We have also used the algorithm to generate the results for the two-factor blocked quadratic designs in Table 1 of Atkinson and Donev (1989), obtaining the same results or slightly better.

The referee suggested that we consider the KL algorithm of Atkinson and Donev (1989), for which Fortran code can be found in Atkinson and Donev (1992). They speeded up the Fedorov algorithm by considering only a fraction of the $n(N-1)$ possible exchanges of design and candidate points. In the notation of this paper, their method consists of ordering the values of $z'z$, and then considering only a fraction, say the smallest quarter, of the points currently in the design for removal, and only say the largest half of the candidate points for inclusion. These fractions were those used in Atkinson and Donev (1989). In our experiments we found that the KL algorithm was between a few per cent and 3.5 times faster than DOPT. However, we found that the KL algorithm usually found designs which were not as good as those found by the DOPT algorithm presented here. The KL algorithm might have produced better results with a different choice for the two fractions.

TABLE 1

Maximum determinants for quadratic surface designs as reported by Galil and Kiefer (1980) and from DOPT

p	n	*Results from the following algorithms:* *Galil and Kiefer*	*DOPT*
4	17	0.1521×10^{14}	0.1529×10^{14}
4	24	0.6566×10^{16}	0.6577×10^{16}
4	25	0.1427×10^{17}	0.1424×10^{17}
5	26	0.1132×10^{24}	0.1168×10^{24}
5	28	0.5930×10^{24}	0.6130×10^{24}
5	29	0.1283×10^{25}	0.1326×10^{25}

Our version of the KL algorithm included the Cook and Nachtsheim (1989) algorithm for swapping design points between blocks.

The use of the Cauchy–Schwarz inequality cuts out some of the pairs which are omitted by the KL algorithm.

We have presented here the basic algorithm. Any competent Fortran programmer should be able to adapt the code to incorporate most of the suggested modifications to the Fedorov algorithm such as those reviewed in Nguyen and Miller (1992).

References

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

——— (1992) *Optimum Experimental Designs*. Oxford: Oxford Scientific.

Cook, R. D. and Nachtsheim, C. J. (1980) A comparison of algorithms for constructing exact D-optimal designs. *Technometrics*, **22**, 315–324.

——— (1989) Computer-aided blocking of factorial and response-surface designs. *Technometrics*, **31**, 339–346.

Donev, A. N. and Atkinson, A. C. (1988) An adjustment algorithm for the construction of exact *D*-optimum experimental designs. *Technometrics*, **30**, 429–433.

Fedorov, V. V. (1969) Theory of optimal experiments. *Preprint 7 LSM*. Moscow State University, Moscow.

——— (1972) *Theory of Optimal Experiments* (eds W. J. Studden and E. M. Klimko). New York: Academic Press.

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

Gentleman, W. M. (1974) Algorithm AS 75: Basic procedures for large, sparse or weighted linear least squares problems. *Appl. Statist.*, **23**, 448–454.

Healy, M. J. R. (1968) Algorithm AS 6: Triangular decomposition of a symmetric matrix. *Appl. Statist.*, **17**, 195–197.

Miller, A. J. (1992) Algorithm AS 274: Least squares routines to supplement those of Gentleman. *Appl. Statist.*, **41**, 458–478.

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

Mitchell, T. J. and Miller, F. L. (1970) Use of "design repair" to construct designs for special linear models. *Report ORNL-4661*, pp. 130–131. Mathematics Division, Oak Ridge National Laboratory, Oak Ridge.

Nachtsheim, C. J. (1987) Tools for computer-aided design of experiments. *J. Qual. Technol.*, **19**, 132–160.

Nguyen, N.-K. and Miller, A. J. (1992) A review of some exchange algorithms for constructing discrete D-optimal designs. *Comput. Statist. Data Anal.*, **14**, 489–498.

SAS Institute (1989) *SAS/QC Software: Reference*, 1st edn, ch. 6. Cary: SAS Institute.

St John, R. C. and Draper, N. R. (1975) D-optimality for regression designs: a review. *Technometrics*, **17**, 15–23.

Van Schalkwyk, D. J. (1971) On the design of mixture experiments. *PhD Thesis*. University of London, London.

Welch, W. J. (1982) Branch-and-bound search for experimental designs based on D-optimality and other criteria. *Technometrics*, **24**, 41–48.

Wichmann, B. A. and Hill, I. D. (1982) Algorithm AS 183: An efficient and portable pseudo-random number generator. *Appl. Statist.*, **31**, 188–190.