IMPERIAL COLLEGE OF SCIENCE AND TECHNOLOGY

Department of Mathematics

# COMPUTER-AIDED DESIGN OF EXPERIMENTS

by

William J. Welch

Thesis submitted for the degree of

Doctor of Philosophy in the University of London

and the

Diploma of Membership of Imperial College

October, 1981

## ABSTRACT

This thesis is concerned with computer-construction of experimental designs. The design problem is to choose a fixed number of design points from a design region of interest in some optimal way. An assumed linear model must also be specified. Various criteria are considered for an optimal choice of the design points, and algorithms are proposed for the automatic construction of designs using these criteria. Throughout, the major computational difficulty stems from the search for exact designs, where only integer-valued replications are allowed.

The first part describes a branch and bound algorithm for construction of D-optimal exact designs. This algorithm, unlike those currently available, can always find an optimal design. It may also be used to produce a catalogue of all designs that are optimal or near-optimal, to be compared by other criteria of interest. The second part is relevant to experiments with the objective of response surface estimation; it considers two criteria based on the variance of the estimated response and suggests an algorithm adapting Mitchell's DETMAX. The third part describes a new criterion, taking account of bias arising from model inadequacy when estimating response surfaces, and two associated algorithms are given. A FORTRAN listing implementing the above algorithms is included in the fourth and final part.

## ACKNOWLEDGEMENTS

I am grateful to my supervisor, Dr. A.C. Atkinson, for his guidance and encouragement throughout the three years of this research. I would also like to thank Prof. D.R. Cox and Dr. H.P. Wynn for several valuable comments.

This work was funded by Warren Spring Laboratory, Stevenage, England. Amongst the many helpful contributions from Warren Spring Laboratory, I am particularly indebted to Mr. R. Derry and Dr. P.G. Christie.

CONTENTS


| | | Page |
|---|---|---|
| INTRODUCTION | | 6 |
| PART 1 | BRANCH AND BOUND SEARCH FOR EXPERIMENTAL DESIGNS BASED UPON D-OPTIMALITY AND OTHER CRITERIA | 9 |
| | 1. Introduction | 10 |
| | 2. A branch and bound algorithm | 12 |
| | 3. Designs for the quadratic response surface model with three factors at three levels | 23 |
| | 4. Conclusions | 24 |
| | References | 25 |
| | Fig. 1 The binary tree generated by the algorithm for the example discussed in section 2.4. | 27 |
| | Table 1 Nodes generated by the algorithm for the example discussed in section 2.4 | 28 |
| | Table 2 Design properties for the quadratic response surface model with three factors at three levels | 29 |
| | Table 3 Design points for the quadratic response surface model with three factors at three levels | 30 |
| PART 2 | COMPUTER-AIDED DESIGN OF EXPERIMENTS FOR RESPONSE SURFACE ESTIMATION | 32 |
| | 1. Introduction | 33 |
| | 2. The algorithm | 35 |
| | 3. An example | 40 |
| | 4. Conclusions | 40 |
| | References | 41 |
| | Table 1 Designs and design properties | 42 |

| | | Page |
|---|---|---|
| PART 3 | A MEAN SQUARE ERROR CRITERION FOR THE DESIGN OF EXPERIMENTS | 44 |
| | 1. Introduction | 45 |
| | 2. A new mean square error criterion | 48 |
| | 3. Design construction | 52 |
| | 4. Conclusions | 59 |
| | Appendix 1 Some computational details | 60 |
| | References | 68 |
| | Table 1 Optimal approximate designs and properties | 70 |
| | Table 2 Optimal exact designs and properties (n = 8) | 71 |
| PART 4 | FORTRAN LISTING OF ALGORITHMS | 73 |
| APPENDIX | ALGORITHMIC COMPLEXITY: THREE NP-HARD PROBLEMS IN COMPUTATIONAL STATISTICS | 127 |
| | 1. Introduction | 128 |
| | 2. NP-completeness and NP-hardness | 128 |
| | 3. Cluster analysis | 131 |
| | 4. Subset selection in regression analysis | 133 |
| | 5. D-optimal exact design of experiments | 133 |
| | 6. Conclusions | 136 |
| | References | 137 |

## INTRODUCTION

The experiments motivating the ideas proposed here are aimed at estimating a yield, observable with random error, over a design region of up to, perhaps, six explanatory variables. The design problem is to choose a number of design points from the design region at which observations on the yield are to be taken. Typically, time and cost constraints limit the number of observations to a small, fixed value. To proceed further with the problem it will also be necessary to specify a model linear in some unknown parameters for the response relationship.

Much attention has been paid in the literature to the construction of D-optimal designs for given design region, number of observations and linear model. This criterion seeks the design that minimizes the determinant of the covariance matrix of the least squares estimators of the parameters. However, when we require an exact design with an integer number of replications at any point in the region of interest, a major computational obstacle is also imposed. The approximate design theory, allowing relaxation of the integer-replication constraint and avoidance of the computational difficulties, will often be inappropriate for experiments with few observations.

Because of the computational difficulties, the currently available algorithms do not necessarily locate a D-optimal design. In Part I a branch and bound method is described that does guarantee optimality, albeit at greater computational expense than existing algorithms such as Mitchell's DETMAX. The branch and bound algorithm can generate a catalogue of all designs that are optimal or near-optimal; they may then be compared by other, secondary, criteria of interest.

When the ultimate objective of the experiment is to estimate not the unknown parameters but the response, it may be better to consider criteria such as G- and V-optimality. The G-optimality criterion chooses a design to minimize the maximum variance of the estimated responses over the design region, whereas V-optimality minimizes the average variance. In contrast to the approximate theory, D- and G-optimality are not equivalent over the set of exact designs. Part 2 shows how Mitchell's DETMAX algorithm may be adapted to the G and V criteria.

All of the above criteria are based on error arising from the sampling variance only. If it is admitted that the assumed linear model may be inadequate, invariably the case in practice, the squared bias of the response estimators should also be considered. However, such mean square error criteria are inherently intractable as they depend on the true responses we are attempting to estimate. The approach described in Part 3 partially overcomes this intractability, chiefly by protecting against mean square error arising from a wide class of non-parametric alternatives to the assumed model. Two algorithms are also presented for constructing designs according to the adopted criterion.

A FORTRAN computer listing incorporating all of the proposed algorithms, with a user's guide, is included in the fourth and final part of the thesis.

The Appendix contains some applications of the theory of algorithmic complexity that are not central to the theme of the thesis. However, one of the examples relates to the construction of D-optimal exact designs.

# PART 1

# BRANCH AND BOUND SEARCH FOR EXPERIMENTAL DESIGNS BASED UPON D-OPTIMALITY AND OTHER CRITERIA

## 1. INTRODUCTION

We are concerned with the problem of choosing n experimental settings or points $x_{(i)}$ $(i = 1, \ldots, n)$, possibly vector-valued and not necessarily distinct, from a specified design region $\mathcal{X}$ at which to take observations $Y_{(i)}$ $(i = 1, \ldots, n)$. In addition, our interest will centre upon the situation where observations are expensive and, hence, n is fixed at a value small enough to render the approximate design theory of Kiefer and Wolfowitz (1959) inappropriate. We assume the linear model

$$Y_{(i)} = \underline{f}\,(x_{(i)})^T \underline{\beta} + \varepsilon_{(i)}, \qquad i = 1, \ldots, n,$$

where $\underline{f}$ is a kxl vector of given linearly independent functions on $\mathcal{X}$ $(k \leq n)$, $\underline{\beta}$ is a kxl vector of unknown parameters and $\varepsilon_{(i)}$ is a random variable for error. It is further assumed that the errors $\varepsilon_{(i)}$ $(i = 1, \ldots, n)$ are independently distributed, each with mean zero and variance $\sigma^2$.

Attention will be restricted to cases where $\mathcal{X}$ is a design region of r candidate sites $x_j$ $(j = 1, \ldots, r)$, thereby excluding continuous regions, since

(i) certain computational advantages ensue, and

(ii) the experimental settings may be qualitative or, if quantitative, the experimenter may wish to limit each factor to only a few levels.

Adapting the notation of Kiefer and Wolfowitz an n-point design assigning $n_j$ points to site $x_j$ $(j = 1, \ldots, r; \sum_{j=1}^{r} n_j = n)$ can be represented by an rxl vector $\underline{p}_n = (p_1, \ldots, p_r)$ where $p_j = n_j n^{-1}$. For the design $\underline{p}_n$ define the kxk matrix

$$\underline{M}(\underline{p}_n) = \sum_{j=1}^{r} p_j \underline{f}\,(x_j)\,\underline{f}\,(x_j)^T,$$

a normalized (for n) version of the familiar $\underline{X}^T\underline{X}$ matrix. Our primary criterion for choosing the design will be that of D-optimality, proposed by Wald (1943) but given the current nomenclature by Kiefer (1958). A D-optimal n-point design or, for brevity, $D_n$-optimal design, $\underline{p}_n^*$ satisfies

$\underset{n}{p}^{*}$ maximizes $\det\{\underline{M}(\underset{n}{p})\}$

subject to $0 \leq p_j \leq 1, \qquad j = 1, \ldots, r \qquad (1)$

$$\sum_{j=1}^{r} p_j = 1, \qquad (2)$$

$p_j = n_j n^{-1}$, $n_j$ an integer, $j = 1, \ldots, r$. (3)

For designs $\underset{n}{p}$ where $\underline{M}(\underset{n}{p})$ is nonsingular and hence the parameters $\underline{\beta}$ are estimable, we also need to define the quantities

$$d(x_j, x_{j'}, \underset{n}{p}) = \underline{f}(x_j)^T \underline{M}^{-1}(\underset{n}{p})\, \underline{f}(x_{j'}), \quad j, j' = 1, \ldots, r$$

and

$$d(x_j, \underset{n}{p}) = d(x_j, x_j, \underset{n}{p}), \quad j = 1, \ldots, r.$$

To aid interpretation of the above definitions, note that $\sigma^2 n^{-1} \underline{M}^{-1}(\underset{n}{p})$ is the covariance matrix of the least squares estimators $\hat{\underline{\beta}}$ for the design $\underset{n}{p}$ and $\sigma^2 n^{-1} d(x_j, x_{j'}, \underset{n}{p})$ represents the covariance between the estimated responses $\underline{f}(x_j)^T \hat{\underline{\beta}}$ and $\underline{f}(x_{j'})^T \hat{\underline{\beta}}$ at $x_j$ and $x_{j'}$ $(j, j' = 1, \ldots, r)$.

With n design points and r candidate sites there are ${}^{r+n-1}C_n$ possible designs (though often some will leave the parameters inestimable) and, hence, one way to construct a $D_n$-optimal design would be to compute and compare all ${}^{r+n-1}C_n$ determinants $\det\{\underline{M}(\underset{n}{p})\}$. Clearly, though, even for moderate n and r, an exhaustive search is computationally infeasible. Box and Draper (1971) perform such a complete enumeration for small factorial experiments, but for larger problems restrict attention to cube and star point designs. An integer programming approach for constrained factorial designs is discussed by Neuhardt and Bradley (1971).

If the n-point condition (3) is removed from the optimization problem discussed above, the maximization is computationally much easier: the algorithms of, for example, Wynn (1970) or Fedorov (1972, pp. 97-104) may be applied, followed by the "rounding" algorithm of Fedorov (1972, p. 157) to recover an n-point design.

A $D_n$-optimal design is not guaranteed, however, and for small n (near k) performance may be poor.

The exchange algorithms of Fedorov (1972, pp. 160-165), Wynn (1972) and Mitchell (1974) are all sequential: a sequence of n-point designs with nondecreasing determinants is generated but, whilst computationally inexpensive compared with exhaustive search, convergence, though assured, may be to a local maximum and a $D_n$-optimal design is again not guaranteed. The balanced array method of Mitchell and Bayne (1978) for fractions of 3-level factorial arrangements may also generate local maxima. Cook and Nachtsheim (1980) review and compare algorithms for constructing $D_n$-optimal designs.

## 2. A BRANCH AND BOUND ALGORITHM

The concept of branch and bound was popularized largely by the algorithm of Little, Murty, Sweeney and Karel (1963) for the travelling salesman problem. We now discuss a branch and bound algorithm for the $D_n$-optimal design problem, with an extension considering other criteria in conjunction with D-optimality.

### 2.1. The binary tree (branching) structure

The optimization problem to maximize $\det\{\underline{M}(p_n)\}$ subject to (1), (2) and (3) may be generalized if the constraint (1) is replaced by

$$\ell_j \le p_j \le u_j, \quad j = 1, \ldots, r, \tag{4}$$

where $\ell_j$ and $u_j$ are specified constants such that $0 \le \ell_j \le u_j \le 1$ $(j = 1, \ldots, r)$. Hereafter, the maximization subject to (2), (3) and (4) for specified $\underline{\ell}$ and $\underline{u}$ will be called a node and for the original maximization, or root node, we merely write $\ell_j = 0$, $u_j = 1$ $(j = 1, \ldots, r)$. Rather than solve the root node directly, though, we may solve two subproblems, or descendant nodes, by choosing the constants $\underline{\ell}$ and $\underline{u}$ of each descendant to divide the set of all possible n-point designs into two partitions. Comparison of the two solutions of the descendants trivially solves the root.

The partitioning or branching process may be continued recursively: for every node, including the root, one of two conditions holds. Either

(i) the constraints (2), (3) and (4) of the node allow one and only one n-point design and the node maximization immediately follows, or

(ii) the constraints admit more than one n-point design, whereupon we select integers $j_o(\underline{\ell}, \underline{u})$ $(1 \le j_o(\underline{\ell}, \underline{u}) \le r)$ and $n_o(\underline{\ell}, \underline{u})$ and create two descendant nodes. The descendants partition the n-point designs allowed by the current node into those with at least $n_o$ design points $x_{j_o}$ and those with less than $n_o$ points $x_{j_o}$ (we omit the dependence of $j_o$ and $n_o$ on $\underline{\ell}$ and $\underline{u}$ for typographical brevity). To effect a division into non-empty partitions we require that

$$1 + \ell_{j_o} n \le n_o \le u_{j_o} n,$$

$$\sum_{\substack{j=1 \\ j \ne j_o}}^{r} \ell_j + n_o n^{-1} \le 1$$

and

$$\sum_{\substack{j=1 \\ j \ne j_o}}^{r} u_j + (n_o - 1) n^{-1} \ge 1.$$

Finally, we update the constraints (4): the descendant nodes take the values of $\underline{\ell}$ and $\underline{u}$ of the current node, except that for the first descendant $\ell_{j_o} = n_o n^{-1}$ and for the second $u_{j_o} = (n_o - 1)n^{-1}$.

Thus, every node has either zero or two descendant nodes and a binary tree is generated, with the ${}^{r+n-1}C_n$ n-point designs located at the extremes. Comparison of their determinants clearly guarantees a $D_n$-optimal design, but this is essentially exhaustive search; indeed, it is an inefficient method of exhaustive search. By introducing bounds, however, much of the tree need not be explicitly evaluated and major computational improvements ensue.

## 2.2 Bounds

Consider a node where the constraints (2), (3) and (4) allow more than one n-point design. We now give two upper bounds for the node maximization.

For the first bound, we need to define

$$n^{(a)} = n \sum_{j=1}^{r} \ell_j,$$

$$\underline{M}(\underline{\ell}, \alpha) = \sum_{j=1}^{r} \left( \ell_j + \frac{\alpha}{nr} \right) \underline{f}(x_j)\, \underline{f}(x_j)^T, \quad \alpha > 0 \tag{5}$$

and

$$d(x_j, \underline{\ell}, \alpha) = \underline{f}(x_j)^T \underline{M}^{-1}(\underline{\ell}, \alpha)\, \underline{f}(x_j),$$

where the definition (5) ensures that the kxk matrix $\underline{M}(\underline{\ell}, \alpha)$ is positive definite, following Mitchell (1974). For small $\alpha > 0$ if $\underline{M}(\underline{\ell}, 0)$ is already positive definite, then the proportional error in using $\det\{\underline{M}(\underline{\ell}, \alpha)\}$ below, instead of $\det\{\underline{M}(\underline{\ell}, 0)\}$, is approximately $\alpha$. In later computations $\alpha$ is set at 0.001. The integer $n^{(a)}$ is the number of points assigned by $\underline{\ell}$; therefore, any design $\underline{p}_n$ satisfying the node constraints is comprised of the $n^{(a)}$ assigned points plus any further $n-n^{(a)}$ points allowed by the constraints. Let these further points be $x_{(i)}$ $(i = n^{(a)} + 1, \ldots, n)$. Now

$$\det\{\underline{M}(\underline{p}_n)\} = \det\{\underline{M}(\underline{\ell}, 0) + n^{-1} \underline{F}\underline{F}^T\}$$

$$\leq \det\{\underline{M}(\underline{\ell}, \alpha) + n^{-1} \underline{F}\underline{F}^T\}$$

and $\underline{F}$ is the kx$(n-n^{(a)})$ matrix $\left[\underline{f}(x_{(n^{(a)}+1)}), \ldots, \underline{f}(x_{(n)})\right]$. Applying Lemma 2.5.1 of Fedorov (1972, pp. 99-100) and the Hadamard inequality for positive definite matrices, it follows that

$$\det\{\underline{M}(\underline{p}_n)\} \leq \det\{\underline{M}(\underline{\ell}, \alpha)\} \det\{\underline{I} + n^{-1} \underline{F}^T \underline{M}^{-1}(\underline{\ell}, \alpha)\, \underline{F}\}$$

$$\leq \det\{\underline{M}(\underline{\ell}, \alpha)\} \prod_{i=n^{(a)}+1}^{n} \{1 + d(x_{(i)}, \underline{\ell}, \alpha) n^{-1}\}$$

$$\leq \det\{\underline{M}(\underline{\ell}, \alpha)\} \{1 + \bar{d}(\underline{\ell}, \alpha)\, n^{-1}\}^{n-n^{(a)}}, \tag{6}$$

where $\bar{d}(\underline{\ell}, \alpha) = \max_{\substack{j=1,\ldots,r \\ \ell_j < u_j}} d(x_j, \underline{\ell}, \alpha)$

and $\underline{I}$ is the $(n-n^{(a)}) \times (n-n^{(a)})$ identity matrix. The bound on the maximization provided by the right-hand side of (6) is denoted by $b_1(\underline{\ell}, \underline{u})$.

Note that if terms involving $\alpha$ are neglected, then (6) is an equality when $n-n^{(a)} = 1$ and the node maximization follows.

The second upper bound on the node maximization is obtained by removing the n-point condition (3) and so relaxing the optimization problem (the relaxed problem cannot yield a lower maximum). The constraints (2) and (4) in isolation define a class of approximate designs, approximate in the sense that $p_j$ $(j = 1, \ldots, r)$ need not correspond to an integer number of observations. Theorem 1 below, an adaptation of Theorem 2.1 of Wynn (1977), suggests a computational method for the relaxed maximization.

Theorem 1. The following two conditions on $\underline{p}^*$ are equivalent.

(i) $\underline{p}^*$ maximizes $\det\{\underline{M}(\underline{p})\}$ subject to (2) and (4).

(ii) $\underline{p}^*$ satisfies (2) and (4) and for all h such that $p_h^* < u_h$ $(h=1, \ldots, r)$ and for all i such that $p_i^* > \ell_i$ $(i=1, \ldots, r)$, $d(x_h, \underline{p}^*) \leq d(x_i, \underline{p}^*)$.

The proof may be derived from that of Wynn and is omitted.

From an arbitrary approximate design $\underline{p}^{(1)}$ satisfying (2) and (4) with $\underline{M}(\underline{p}^{(1)})$ nonsingular we iteratively generate a sequence of designs $\underline{p}^{(s)}$ $(s = 1,2,\ldots)$. At iteration s let h and i satisfy

$$d(x_h, \underline{p}^{(s)}) = \max_{\substack{j=1,\ldots,r \\ p_j^{(s)} < u_j}} d(x_j, \underline{p}^{(s)})$$

and

$$d(x_i, \underline{p}^{(s)}) = \min_{\substack{j=1,\ldots,r \\ p_j^{(s)} > \ell_j}} d(x_j, \underline{p}^{(s)}).$$

Then $\underline{p}^{(s+1)}$ is derived from $\underline{p}^{(s)}$ as

$$p_j^{(s+1)} = \begin{cases} p_j^{(s)}, & j = 1, \ldots, r; \ j \neq h, i \\ p_j^{(s)} + \delta^{(s)}, & j = h \\ p_j^{(s)} - \delta^{(s)}, & j = i. \end{cases}$$

It may be shown that

$$\det\{\underline{M}(\underline{p}^{(s+1)})\} = (1+\Delta)\det\{\underline{M}(\underline{p}^{(s)})\}$$

where

$$\Delta = \delta^{(s)}\{d(x_h, \underline{p}^{(s)}) - d(x_i, \underline{p}^{(s)})\}$$

$$- (\delta^{(s)})^2\{d(x_h, \underline{p}^{(s)})\, d(x_i, \underline{p}^{(s)}) - d^2(x_h, x_i, \underline{p}^{(s)})\}.$$

By differentiating $\Delta$ with respect to $\delta^{(s)}$ and equating with zero we find that to maximize $\det\{\underline{M}(\underline{p}^{(s+1)})\} / \det\{\underline{M}(\underline{p}^{(s)})\}$ and satisfy the constraints imposed by $u_h$ and $\ell_i$

$$\delta^{(s)} = \min\left[u_h - p_h^{(s)},\ p_i^{(s)} - \ell_i,\ \frac{d(x_h, \underline{p}^{(s)}) - d(x_i, \underline{p}^{(s)})}{2\{d(x_h, \underline{p}^{(s)})d(x_i, \underline{p}^{(s)}) - d^2(x_h, x_i, \underline{p}^{(s)})\}}\right].$$

Clearly, $\det\{\underline{M}(\underline{p}^{(s+1)})\} \geq \det\{\underline{M}(\underline{p}^{(s)})\}$ since equality can be achieved if $\delta^{(s)} = 0$. Furthermore, by Theorem 1, the monotonic nondecreasing sequence of determinants $\det\{\underline{M}(\underline{p}^{(s)})\}$ converges to $\det\{\underline{M}(\underline{p}^{*})\}$, an upper bound on the nodal maximization also incorporating (3) which we denote by $b_2(\underline{\ell}, \underline{u})$.

For a node in the binary tree where the constraints allow more than one n-point design, we seek to avoid creating the descendants by employing the bounds $b_1(\underline{\ell}, \underline{u})$ and $b_2(\underline{\ell}, \underline{u})$. Let $D^*$ represent the maximum determinant $\det\{\underline{M}(\underline{p}_n)\}$ of those n-point designs encountered by the algorithm prior to consideration of the current node. If

$$\min\{b_1(\underline{\ell}, \underline{u}),\ b_2(\underline{\ell}, \underline{u})\} < D^* \qquad (7)$$

holds, then all n-point designs allowed by the node constraints have a determinant inferior to $D^*$. Therefore, we need not consider any descendants of the current node and do not proceed with the branching operation.

The bound $b_2(\underline{\ell}, \underline{u})$ is computed as the limit of a sequence, but in practice we are only concerned whether $b_2(\underline{\ell}, \underline{u}) < D^*$ or $b_2(\underline{\ell}, \underline{u}) \geq D^*$ [assume that $b_1(\underline{\ell}, \underline{u}) \geq D^*$]. At iteration s, if $\det\{\underline{M}(\underline{p}^{(s)})\} \geq D^*$ then, clearly, the second case is true and the sequence is terminated. However, if $\det\{\underline{M}(\underline{p}^{(s)})\} < D^*$ then, rather than invoke an arbitrary stopping rule, we employ Theorem 2 below, the proof of which closely follows Kiefer (1961).

Theorem 2. For an approximate design $\underline{p}$ satisfying (2) and (4) the inequality

$$b_2(\underline{\ell}, \underline{u}) \leq \det\{\underline{M}(\underline{p})\} \exp\{d^*(\underline{p}) - k\}$$

holds, where

$$d^*(\underline{p}) = \max_{\underline{p}' \text{ satisfies (2) and (4)}} \sum_{j=1}^{r} p_j' \, d(x_j, \underline{p}) .$$

To compute the vector $\underline{p}'$ providing $d^*(\underline{p})$ we initially set $p_j' = \ell_j$ $(j = 1, \ldots, r)$ to satisfy the lower bounds in (4). The remaining $1 - \sum_{j=1}^{r} \ell_j$ is then assigned by increasing the elements $p_j'$ subject to the upper bounds in (4) in order of decreasing $d(x_j, \underline{p})$ $(j = 1, \ldots, r)$ until $\sum_{j=1}^{r} p_j' = 1$. The implication of Theorem 2 is that when $\det\{\underline{M}(\underline{p}^{(s)})\} < D^*$

(i) if $\det\{\underline{M}(\underline{p}^{(s)})\} \exp\{d^*(\underline{p}^{(s)}) - k\} < D^*$ then $b_2(\underline{\ell}, \underline{u}) < D^*$ and the sequence may be terminated, whereas

(ii) if $\det\{\underline{M}(\underline{p}^{(s)})\} \exp\{d^*(\underline{p}^{(s)}) - k\} \geq D^*$ we continue iterating.

2.3 A strategy for branching

The strategy proposed for choosing $j_o$ and $n_o$ in the branching operation is based upon empirical studies only. We use the heuristics

$$d(x_{j_o}, \underline{\ell}, \alpha) = \max_{\substack{j=1,\ldots,r \\ \ell_j<u_j}} d(x_j, \underline{\ell}, \alpha) \tag{8}$$

and

$$n_o = \ell_{j_o} n + 1$$

to select $j_o$ and $n_o$. Thus, we attempt to branch such that the descendant with increased lower constraint $\ell_{j_o}$ is favoured relative to the descendant with decreased upper constraint $u_{j_o}$; similar strategies have been found effective in integer programming [see, for example, Garfinkel and Nemhauser (1972)]. The rule for selecting $n_o$ assumes that we are interested in design problems where n - k is small.

If branching takes place the next node to be evaluated is always the descendant with increased lower constraint $\ell_{j_o}$, and the other descendant is placed on the top of a stack of nodes awaiting attention. Branching does not take place if the bound (7) applies or the node optimization is solved (and $D^*$ is updated if necessary), a case arising when one and only one n-point design is defined by the node constraints or $n-n^{(a)} = 1$. When nodes with no descendants occur, the next node to be processed is removed from the top of the stack. The algorithm commences with an empty stack, $D^*$ is initialized at zero, and the current node of interest is the root. Termination occurs when a node with no descendants is encountered and the stack is empty, and the

design or designs associated with $D^*$ at termination are the $D_n$-optimal designs.

2.4 An example

Let n=3 and let the model be $E(Y_{(i)}) = \beta_1+\beta_2 x_{(i)}$ (i = 1, 2, 3) with design region comprising the r=3 candidates $x_1 = -1$, $x_2 = 0$, and $x_3 = 1$. Kiefer (1961) shows that the D-optimal 3-point designs are $\underline{p}_3^{*(1)} = (\frac{1}{3}, 0, \frac{2}{3})$ and $\underline{p}_3^{*(2)} = (\frac{2}{3}, 0, \frac{1}{3})$, but we use this simple example to illustrate the branch and bound procedure.

The 5 nodes generated by the algorithm are listed in Table 1 in the order of their consideration and Figure 1 illustrates the binary tree arrangement. We set $\alpha$ at 0.001 throughout. At node 1, the root, the inequality (7) does not hold and the branching operation takes place. For this node the heuristic (8) yields a tie: $d(x_j, \underline{\ell}, \alpha)$ achieves a maximum at $x_1$ and $x_3$ but we arbitrarily choose $x_3$ for branching. Hence, node 2 with $\ell_3 = 1/3$ is constrained to have at least one design point $x_3$, whereas the other descendant, node 5, with $u_3 = 0$ allows only those 3-point designs with zero design points $x_3$. Similarly, branching takes place at node 2, but this time the candidate $x_1$ is indicated by (8). At node 3, since $n-n^{(a)} = 1$, if terms involving $\alpha$ are neglected we have equality in the bound (6) and $b_1(\underline{\ell}, \underline{u})$ may be used to represent the maximum determinant of all 3-point designs satisfying the node constraints, with proportional error approximately 0.001. The designs providing the maximum for this node immediately follow as $\bar{d}(\underline{\ell}, \alpha)$ in (6) is achieved for both $x_1$ and $x_3$ and either may be added to the 2 points already assigned by $\underline{\ell}$, yielding the designs $\underline{p}_3^{*(1)}$ and $\underline{p}_3^{*(2)}$ above. For nodes 4 and 5 the inequality (7) holds, $D^*$ is never superseded again, and the algorithm demonstrates that $\underline{p}_3^{*(1)}$ and $\underline{p}_3^{*(2)}$ are, in fact, the $D_3$-optimal designs.

## 2.5 Exploiting symmetries

It may be possible to exploit symmetries when branching to impose tighter constraints on one of the descendant nodes and reduce the size of the binary tree. For a given ancestor, if there exist a kxk matrix $\underline{A}$ with $\{\det(\underline{A})\}^2 = 1$ and a permutation $\Pi(1), \ldots, \Pi(r)$ of the integers $1, \ldots, r$ such that

$$\underline{f}(x_j) = \underline{A}\,\underline{f}(x_{\Pi(j)}), \qquad j = 1, \ldots, r \tag{9}$$

$$\ell_j = \ell_{\Pi(j)}, \qquad j = 1, \ldots, r \tag{10}$$

and

$$u_j = u_{\Pi(j)}, \qquad j = 1, \ldots, r \tag{11}$$

then we say that the ancestor is symmetric with respect to $\Pi$. For example, node 1 in Table 1 is symmetric with respect to $\Pi(1) = 3$, $\Pi(2) = 2$, $\Pi(3) = 1$ and

$$\underline{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Consider an n-point design $\underline{p}_n$ satisfying the ancestor node constraints (2), (3) and (4). The design $\underline{p}_n^{(\Pi)} = (p_{\Pi(1)}, \ldots, p_{\Pi(r)})$ also satisfies (2), (3) and (4) because of (10) and (11), and some simple rearrangement and the application of (9) yield the result

$$\det\{\underline{M}(\underline{p}_n)\} = \det\{\underline{M}(\underline{p}_n^{(\Pi)})\}. \tag{12}$$

When partitioning on $x_{j_o}$ to form the two descendants suppose that $\Pi(j_o) \neq j_o$. We may now decrease two elements of $\underline{u}$ rather than just one to create the second descendant: $u_{j_o}$ and $u_{\Pi(j_o)}$ are both reduced to $(n_o - 1)n^{-1}$. To see that a $D_n$-optimal design is still guaranteed, consider branching on $j_o$ as before (ignoring the symmetry) and then partitioning the second descendant by further branching on $\Pi(j_o)$. The three resulting descendants take the constraints of the ancestor with the following adjustments:

D1 $\ell_{j_o}$ increased

D2.1 $u_{j_o}$ decreased, $\ell_{\pi(j_o)}$ increased

D2.2 $u_{j_o}$ decreased, $u_{\pi(j_o)}$ decreased.

Because of the symmetry of increasing the lower constraint at either $j_o$ or $\pi(j_o)$ it may be shown that any n-point design permitted by descendant D2.1 is only a permutation $\underline{p}_n^{(\pi)}$ of a design $\underline{p}_n$ in D1 with the same determinant by (12). Hence D2.1 may be ignored and the algorithm continues with the two descendant nodes D1 and D2.2. In general, we may tighten t extra constraints on the second descendant if t permutations $\pi_1, \ldots, \pi_t$ and t kxk matrices $\underline{A}_1, \ldots, \underline{A}_t$ exist such that $\{\det(\underline{A}_i)\}^2 = 1$ $(i = 1, \ldots, t)$, $\pi_i$ and $\underline{A}_i$ satisfy (9), (10) and (11) $(i = 1, \ldots, t)$ and $j_o, \pi_1(j_o), \ldots, \pi_t(j_o)$ are distinct.

## 2.6 Augmenting designs

In the design augmentation problem we wish to add a further $n^{(1)}$ points to $n^{(0)}$ points at which observations have already been taken, to produce a D-optimal $(n^{(0)}+n^{(1)})$-point design subject to inclusion of the first $n^{(0)}$ points.

Only the specification of the root node need be modified: we now set $\ell_j = n_j^{(0)} (n^{(0)} + n^{(1)})^{-1}$ $(j = 1, \ldots, r)$ where $n_j^{(0)}$ observations are allocated to $x_j$ by the initial $n^{(0)}$ points and proceed recursively as before.

## 2.7 Multiple criteria

Whilst not explicitly stated, the above branch and bound algorithm will not only guarantee a $D_n$-optimal design, but will provide a complete catalogue of all $D_n$-optimal designs if ties occur (it is computationally simple to recover any optimal designs lost due to exploitation of symmetries when the algorithm terminates). The major criticism, however, must surely be that there is over-reliance on a single criterion, D-optimality.

Two further criteria that might be considered in conjunction with D-optimality when assessing an n-point design $p_n$ are the normalized maximum variance of prediction

$$v_{max}(p_n) = \max_{j=1,\ldots,r} d(x_j, p_n)$$

and the normalized average variance of prediction

$$v_{ave}(p_n) = \frac{1}{r} \sum_{j=1}^{r} d(x_j, p_n).$$

To incorporate these criteria, or any others proposed by the experimenter, we may change our strategy to extend the catalogue of $D_n$-optimal designs to include all those that are "nearly" $D_n$-optimal in some controlled way. For example, for fixed $\gamma$ $(0 < \gamma < 1)$ acceptable

to the experimenter, we could generate all designs $\underline{p}_n$ with $\det\{\underline{M}(\underline{p}_n)\} \geq (1-\gamma)\det\{\underline{M}(\underline{p}_n^*)\}$ by modifying the bounding rule (7) to prevent branching if

$$\min\{b_1(\underline{\ell},\underline{u}),\ b_2(\underline{\ell},\underline{u})\} < (1-\gamma)\, D^*.$$

It should be emphasized that on termination we know exactly how far any "nearly" $D_n$-optimal design is from global optimality and hence the algorithm is quite distinct from others not guaranteeing a global maximum. More importantly, though, an extensive catalogue of all designs acceptable by the D-optimality criterion can be produced, which may be compared using the other criteria of interest.

## 3. DESIGNS FOR THE QUADRATIC RESPONSE SURFACE MODEL WITH THREE FACTORS AT THREE LEVELS

In this section some previously unknown $D_n$-optimal designs are presented and the extension of the algorithm to multiple criteria is illustrated.

For the design region we assume the $3^3$ factorial arrangement where each of the 3 factors or explanatory variables may take one of 3 levels, coded as 0, 1 and 2. The quadratic response surface model for 3 factors has 10 parameters and may be written as

$$E(Y_{(i)}) = \beta_o + \sum_{s=1}^{3} (\beta_s x_{(i)s} + \beta_{ss} x_{(i)s}^2) + \sum_{s=1}^{2} \sum_{t=s+1}^{3} \beta_{st} x_{(i)s} x_{(i)t}, \qquad i=1,\ldots,n,$$

where $(x_{(i)1}, x_{(i)2}, x_{(i)3})$ are the 3 components of $\underline{x}_{(i)}$ $(i = 1,\ldots,n)$. The cases $n = 10,\ldots, 20$ are enumerated. This choice of design region and model yields a set of design problems within the computer time limitations of the present algorithm, but still of sufficient computational complexity that the advantages are apparent.

In Table 2 the $D_n$-optimal designs are compared with a "compromise" design which seeks to improve performance as measured by the maximum and, to a lesser extent, average variances of prediction for a small loss in D-optimality efficiency. The design points are given in Table 3; where other $D_n$-optimal designs exist they differ only by a symmetry such as relabelling of the three factors.

The Mitchell and Bayne (1976, 1978) catalogue of designs based upon D-optimality has recently been improved by Galil and Kiefer (1980) using a modified version of Mitchell's DETMAX algorithm. Comparison of the updated catalogue with the designs supplied here shows that modified DETMAX has found a $D_n$-optimal design in each case, though their optimality had not previously been proved. Assuming that D-optimality is not the sole concern of the experimenter, however, we now look at the case n = 18 in detail. The compromise design achieves minimum $v_{max}(p_{18})$ and minimum $v_{ave}(p_{18})$ amongst the class of designs such that $\det\{M(p_{18})\} \geq 0.95 \det\{M(p_{18}^*)\}$ and substantially reduces the value of $v_{max}$ achieved by the $D_{18}$-optimal design. The enumeration of this case required about 63 seconds of CDC 6600 computer time and 2531 nodes were generated for a problem with approximately $1.03 \times 10^{12}$ possible designs. As n and r increase so the number of possible designs, ${}^{r+n-1}C_n$, increases exponentially and, whilst experience suggests that the number of nodes requiring evaluation grows more slowly than ${}^{r+n-1}C_n$, the execution time rapidly becomes prohibitive.

## 4. CONCLUSIONS

A branch and bound algorithm guaranteeing a catalogue of all $D_n$-optimal designs was presented, with an extension to improve performance by other criteria for a small loss in D-optimality efficiency. In the extended mode we are, in effect, using the algorithm to reduce the class of all possible n-point designs to a smaller class "acceptable" by the criterion of D-optimality. Amongst the reduced class a design may be chosen using secondary criteria. For moderate-sized problems, of computational complexity similar to those enumerated herein, it appears that advantages large enough to be of practical benefit may

follow, compared with previous methods. Further work would be helpful, however, either in improving the branching strategy or tightening the bounds, to enable efficient enumeration of larger problems.

A FORTRAN listing is available upon request to the author.

REFERENCES

BOX, M.J. and DRAPER, N.R. (1971). Factorial designs, the $|X'X|$ criterion, and some related matters. Technometrics, 13, 731-742.

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

FEDOROV, V.V. (1972). Theory of optimal experiments. Translated and edited by W.J. Studden and E.M. Klimko, Academic Press, New York.

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.

GARFINKEL, R.S. and NEMHAUSER, G.L. (1972). Integer programming. Wiley, New York.

KIEFER, J. (1958). On the nonrandomized optimality and randomized nonoptimality of symmetrical designs. Ann. Math. Statist., 29, 675-699.

KIEFER, J. (1961). Optimum experimental designs V, with applications to systematic and rotatable designs. Proceedings of the Fourth Berkeley Symposium on Mathematical Statist. and Probability, 1, 381-405.

KIEFER, J. and WOLFOWITZ, J. (1959). Optimum designs in regression problems. Ann. Math. Statist., 30, 271-294.

LITTLE, J.D.C., MURTY, K.G., SWEENEY, D.W. and KAREL, C. (1963). An algorithm for the travelling salesman problem. Operations Research, 11, 972-989.

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

MITCHELL, T.J. and BAYNE, C.K. (1976). D-optimal fractions of three-level factorial designs. ORNL/CSD-19. National Technical Information Service, U.S. Dept. of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

MITCHELL, T.J. and BAYNE, C.K. (1978). D-optimal fractions of three-level factorial designs. Technometrics, 20, 369-380.

NEUHARDT, J.B. and BRADLEY, H.E. (1971). On the selection of multifactor experimental arrangements with resource constraints. J. Amer. Statist. Assoc., 66, 618-621.

WALD, A. (1943). On the efficient design of statistical investigations. Ann. Math. Statist., 14, 134-140.

WYNN, H.P. (1970). The sequential generation of D-optimum experimental designs. Ann. Math. Statist., 41, 1655-1664.

WYNN, H.P. (1972). Results in the theory and construction of D-optimum experimental designs. J. Roy. Statist. Soc., B34, 133-147.

WYNN, H.P. (1977). Optimum designs for finite populations sampling. In Statistical decision theory and related topics, II. Edited by S.S. Gupta and D.S. Moore, Academic Press, New York.

Figure 1. The binary tree generated by the algorithm for the example discussed in section 2.4.

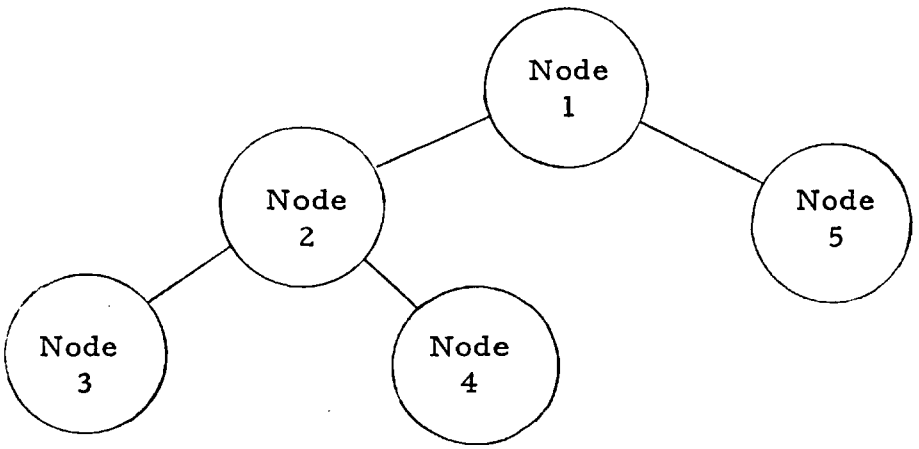

Table 1. Nodes generated by the algorithm for the example discussed in section 2.4

| Node | Ancestor node | Constraints | | | | | | $b_1(\underline{\ell},\underline{u})^+$ | $b_2(\underline{\ell},\underline{u})$ | $D^*$ | Action |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\ell_1$ | $\ell_2$ | $\ell_3$ | $u_1$ | $u_2$ | $u_3$ | | | | |
| 1 | - | 0 | 0 | 0 | 1 | 1 | 1 | .1159E+04 | .1000E+01 | 0 | Branch on $x_3$ |
| 2 | 1 | 0 | 0 | $\frac{1}{3}$ | 1 | 1 | 1 | .1068E+04 | .1000E+01 | 0 | Branch on $x_1$ |
| 3 | 2 | $\frac{1}{3}$ | 0 | $\frac{1}{3}$ | 1 | 1 | 1 | .8894E+00 | .1000E+01 | 0 | $D^*$ replaced by .8894E+00 |
| 4 | 2 | 0 | 0 | $\frac{1}{3}$ | 0 | 1 | 1 | .6695E+02 | .2500E+00 | .8894E+00 | Terminate this branch |
| 5 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | .1159E+04 | .2500E+00 | .8894E+00 | Terminate this branch |

+ $\alpha$ set at 0.001

Table 2. Design properties for the quadratic response surface model with three factors at three levels

| n | $D_n$-optimal designs | | | Compromise design[+] | | |
|---|---|---|---|---|---|---|
| | det($\underline{M}$) | $v_{max}$ | $v_{ave}$ | det($\underline{M}$) | $v_{max}$ | $v_{ave}$ |
| 10 | 1.33E-04 | 27.5 | 13.0 | 1.33E-04 | 27.5 | 13.0 |
| | 1.33E-04 | 34.4 | 14.0 | | | |
| 11 | 3.24E-04 | 16.5 | 12.4 | | | |
| 12 | 3.39E-04 | 17.9 | 11.9 | 3.39E-04 | 17.9 | 11.9 |
| | 3.39E-04 | 18.0 | 11.8 | | | |
| 13 | 4.33E-04 | 15.4 | 10.3 | | | |
| 14 | 4.54E-04 | 11.2 | 9.9 | | | |
| 15 | 4.20E-04 | 12.9 | 10.5 | 4.10E-04 | 11.9 | 10.3 |
| 16 | 4.10E-04 | 13.5 | 10.4 | 3.85E-04 | 12.6 | 10.5 |
| 17 | 4.13E-04 | 14.2 | 10.5 | 3.76E-04 | 13.3 | 10.7 |
| 18 | 4.28E-04 | 14.7 | 10.6 | 4.18E-04 | 12.9 | 10.3 |
| 19 | 4.54E-04 | 12.4 | 10.4 | | | |
| 20 | 4.63E-04 | 12.1 | 10.2 | | | |

[+] As $D_n$-optimal design if no properties appear

Table 3. Design points for the quadratic response surface model with three factors at three levels.

| n | Design[+] | Design points |
|---|---|---|
| 10 | D, C | 002 010 021 101 112 200 202 211 220 222 |
| | D | 000 002 021 101 110 200 202 211 220 222 |
| 11 | D, C | 000 002 020 022 110 121 200 202 211 220 222 |
| 12 | D, C | 000 002 011 020 022 101 110 122 200 202 220 222 |
| | D | 000 002 011 020 022 101 110 112 200 202 220 222 |
| 13 | D, C | 000 002 011 020 022 101 112 120 200 202 210 221 222 |
| 14 | D, C | 000 002 011 020 022 101 110 112 121 200 202 211 220 222 |
| 15 | D | 000 002 011 012 020 022 101 110 122 200 202 212 220 221 222 |
| | C | 000 000 002 011 020 022 101 110 112 121 200 202 211 220 222 |
| 16 | D | 000 002 011 020 022 101 110 122 200 201 202 210 212 220 221 222 |
| | C | 000 000 002 011 020 020 022 101 110 112 121 200 202 211 220 222 |
| 17 | D | 000 001 002 010 012 020 022 100 102 110 121 200 202 211 220 222 222 |
| | C | 000 002 011 020 020 022 022 101 110 112 121 200 202 211 220 222 222 |
| 18 | D | 000 002 002 011 020 022 022 100 112 120 121 200 201 202 210 220 221 222 |
| | C | 000 001 002 010 012 020 022 100 102 111 121 200 201 202 210 212 220 222 |
| 19 | D, C | 000 000 002 012 020 021 022 102 111 120 122 200 201 202 210 212 220 221 222 |
| 20 | D, C | 000 001 002 010 012 020 021 022 100 102 110 121 200 201 202 211 220 220 222 222 |

[+] D = $D_n$ - optimal, C = Compromise.

# PART 2

# COMPUTER-AIDED DESIGN OF EXPERIMENTS FOR RESPONSE SURFACE ESTIMATION

## 1. INTRODUCTION

Although much attention has been paid to the construction of D-optimal exact (integer replication) experiments, criteria concerned with the variance of the response surface estimator such as G-optimality have received scant consideration in the literature. This is inappropriate since, in practice, many experiments are aimed at the estimation of the response surface over a design region of interest rather than estimation of the parameters in the assumed linear model. Moreover, D- and G-optimality are not equivalent for exact designs. An algorithm is proposed for two response estimation design criteria.

Suppose the experimenter wishes to estimate a response y depending on q explanatory variables $\underset{\sim}{x}$ over a design region of r points $\underset{\sim}{x}_1, \ldots, \underset{\sim}{x}_r$, denoted by $\mathcal{X}$. Where the original design region is continuous, for example, a cube or sphere, then a finite grid of r points can be used as an approximation. The design problem is to choose n not necessarily distinct points $\underset{\sim}{x}_{(1)}, \ldots, \underset{\sim}{x}_{(n)}$ from $\mathcal{X}$ at which to take observations $Y_{(1)}, \ldots, Y_{(n)}$ of the response according to an assumed linear model

$$Y_{(i)} = \underset{\sim}{f}(\underset{\sim}{x}_{(i)})^T \underset{\sim}{\beta} + \varepsilon_{(i)} \qquad i = 1, \ldots, n$$

The kx1 vectors $\underset{\sim}{f}$ and $\underset{\sim}{\beta}$ are k specified linearly independent functions on $\mathcal{X}$ and the associated k unknown parameters ($k \leq n$). The random errors $\varepsilon_{(1)}, \ldots, \varepsilon_{(n)}$ are assumed to be uncorrelated, each with mean zero and constant variance $\sigma^2$. Typically, in response surface work, the model will be a first or second order polynomial.

It is convenient to represent an n-point design by an rx1 vector $\underset{\sim}{p}$ where $p_j = n_j n^{-1}$ ($j = 1, \ldots, r$) and $n_j$ is the number of replications at $\underset{\sim}{x}_j$ in $\mathcal{X}$. The D-optimality criterion chooses $\underset{\sim}{p}$ to maximize the determinant of the information matrix

$$\underset{\sim}{M}(\underset{\sim}{p}) = \sum_{j=1}^{r} p_j \, \underset{\sim}{f}(\underset{\sim}{x}_j) \, \underset{\sim}{f}(\underset{\sim}{x}_j)^T$$

or equivalently to minimize the determinant of $\underset{\sim}{M}(\underset{\sim}{p})^{-1}$, the covariance matrix of the least squares estimators $\underset{\sim}{\hat{\beta}}$ normalized for n and $\sigma^2$.

Throughout, we shall assume $\underset{\sim}{M}(\underset{\sim}{p})$ is invertible; singularity is easily avoided by the adjustment of Mitchell (1974).

In the experiments of interest to the author interest instead centres on the response estimators $\hat{y}(\underset{\sim}{x}_j) = \underset{\sim}{f}(\underset{\sim}{x}_j)^T \hat{\underset{\sim}{\beta}}$ $(j = 1, \ldots, r)$. The normalized variance of $\hat{y}(\underset{\sim}{x}_j)$ is

$$d(\underset{\sim}{x}_j, \underset{\sim}{p}) = \underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{M}(\underset{\sim}{p})^{-1} \underset{\sim}{f}(\underset{\sim}{x}_j) \qquad (j = 1, \ldots, r)$$

and we might choose $\underset{\sim}{p}$ to minimize the maximum variance

$$d_{max}(\underset{\sim}{p}) = \max_{j=1,\ldots,r} d(\underset{\sim}{x}_j, \underset{\sim}{p}) \tag{1}$$

or to minimize the average variance

$$d_{ave}(\underset{\sim}{p}) = r^{-1} \sum_{j=1}^{r} d(\underset{\sim}{x}_j, \underset{\sim}{p}) \tag{2}$$

For a weighted average, only minor changes need be made below. The criterion (1) is commonly called G-optimality and we shall call (2) V-optimality, borrowing V from Box and Draper (1959). The mean square error approach of Box and Draper is not pursued here, however, since the number of observations is assumed to be small and the importance of variance relative to bias is supposed large (see Kiefer, 1975 and Galil and Kiefer, 1977).

In the approximate design theory of Kiefer and Wolfowitz (1960), where the design is not constrained to have integer replications, the D-optimality and G-optimality criteria are equivalent. However, when n is small relative to k the approximate theory is inappropriate and there is no such equivalence over the set of exact designs. Furthermore, the constraint of integer replications leads to considerable computational difficulties. For D-optimality, the DETMAX algorithm of Mitchell (1974) with computational improvements by Galil and Kiefer (1980) provides designs at least close to optimality. The branch and bound algorithm of Welch (1981) guarantees, at greater cost, a catalogue of all designs with $\det\{\underset{\sim}{M}(\underset{\sim}{p})\}$ within a specified small tolerance of the optimal design, which can then

be compared using other criteria such as G- and V-optimality. However, the primary design criterion is still D-optimality. The algorithm described below is an adaptation of Mitchell's DETMAX to find G- or V-optimal designs. By means of an example it will be shown that appreciable improvements as measured by these criteria can result compared with the D-optimal design.

## 2. THE ALGORITHM

### 2.1 Excursions

The proposed algorithm can be applied to an arbitrary design criterion minimize $c(\underset{\sim}{p})$ subject to

$$\ell_j \leq p_j \leq u_j \qquad j = 1, \ldots, r$$

where the introduction of constraints allows, for instance, design augmentation. Often, though,we will set $\ell_j = 0$ and $u_j = 1$ $(j = 1, \ldots, r)$. For G- and V-optimality we equate $c(\underset{\sim}{p})$ with $d_{max}(\underset{\sim}{p})$ and $d_{ave}(\underset{\sim}{p})$ in (1) and (2) respectively. For D-optimality, $c(\underset{\sim}{p}) = \det \{ \underset{\sim}{M}(\underset{\sim}{p})^{-1} \}$ and the DETMAX algorithm is recovered.

As in DETMAX, we attempt to improve an initial random n-point design by a series of excursions. An excursion starts by adding or subtracting a point from the current n-point design $\underset{\sim}{p}_1$,and then performs a number of additions or subtractions of a single point, eventually returning to a possibly new n-point design $\underset{\sim}{p}_2$. If $c(\underset{\sim}{p}_2) < c(\underset{\sim}{p}_1)$, then the excursion has succeeded and $\underset{\sim}{p}_2$ is used as the start for the next excursion, whereas if $c(\underset{\sim}{p}_2) \geq c(\underset{\sim}{p}_1)$ a failure has occurred and $\underset{\sim}{p}_1$ is again the starting design for the next attempt. At each step within an excursion two decisions are made: whether to add or subtract a point and then which design point to add or subtract accordingly. The rules for the first decision follow DETMAX and are described by Mitchell (1974) and Galil and Kiefer (1980).

## 2.2 Choosing a point

The algorithm for a criterion $c(\underset{\sim}{p})$ deviates from DETMAX in the choice of a promising point to add or subtract as required. At any stage of an excursion consider adjusting the current n'-point design $\underset{\sim}{p}$ by increasing or decreasing element j by $n^{-1}$. The obvious generalization of DETMAX is to add a point $\underset{\sim}{x}_{j_o}$ satisfying

$$c(\underset{\sim}{p} + n^{-1}\,\underset{\sim}{\delta}_{j_o}) = \min_{j:p_j<u_j} c(\underset{\sim}{p} + n^{-1}\underset{\sim}{\delta}_j) \tag{3}$$

or, when subtracting, select $j_o$ to satisfy

$$c(\underset{\sim}{p} - n^{-1}\underset{\sim}{\delta}_{j_o}) = \min_{j:p_j>\ell_j} c(\underset{\sim}{p} - n^{-1}\underset{\sim}{\delta}_j) \tag{4}$$

where $\underset{\sim}{\delta}_j$ is the rx1 vector with unity in position j and zero elsewhere.

In applying this to the G- and V-optimality criteria we need to define the normalized covariance between $\hat{y}(\underset{\sim}{x}_i)$ and $\hat{y}(\underset{\sim}{x}_j)$

$$d(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p}) = \underset{\sim}{f}(\underset{\sim}{x}_i)^T\, \underset{\sim}{M}(\underset{\sim}{p})^{-1}\, \underset{\sim}{f}(\underset{\sim}{x}_j) \tag{5}$$

and use the result

$$\underset{\sim}{M}(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_j)^{-1} = \underset{\sim}{M}(\underset{\sim}{p})^{-1} - \frac{\underset{\sim}{M}(\underset{\sim}{p})^{-1}\underset{\sim}{f}(\underset{\sim}{x}_j)\underset{\sim}{f}(\underset{\sim}{x}_j)^T\underset{\sim}{M}(\underset{\sim}{p})^{-1}}{d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n} \tag{6}$$

It follows from (5) and (6) that the normalized variance of $\hat{y}(\underset{\sim}{x}_i)$ after a change at $\underset{\sim}{x}_j$ is

$$d(\underset{\sim}{x}_i, \underset{\sim}{p}\pm n^{-1}\underset{\sim}{\delta}_j) = d(\underset{\sim}{x}_i, \underset{\sim}{p}) - \frac{d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})}{d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n}$$

Accordingly, for G-optimality we would compute the minimum in (3) or (4) as appropriate using

$$c(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_j) = \max_{i=1,\ldots,r} \left\{ d(\underset{\sim}{x}_i, \underset{\sim}{p}) - \frac{d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})}{d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n} \right\} \tag{7}$$

whereas for V-optimality

$$c(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_j) = c(\underset{\sim}{p}) - \frac{\sum_{i=1}^{r} d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})}{r\{d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n\}} \tag{8}$$

## 2.3 Random initial designs

As with DETMAX, the generalized algorithm does not guarantee a design minimizing $c(\underset{\sim}{p})$ as it may produce a local optimum at termination. Following Mitchell (1974) the stopping rule abandons further searching when the excursions are forced to n + 6 or n - 6 points. By making repeated attempts from a number of random initial designs, empirical studies not reported here indicate that the best design constructed using G- or V-optimality will usually perform reasonably well, as measured by these criteria, relative to the D-optimal design. The strategy of choosing a random number of points $n_{ran}$ between 1 and k/2, as proposed by Galil and Kiefer (1980), was adopted here to provide a starting design. By sequentially adding a further $n-n_{ran}$ points according to (3) the initial n-point design is produced.

## 2.4 Some computational considerations

For both G- and V-optimality it is necessary to maintain $\underset{\sim}{M}(\underset{\sim}{p})^{-1}$ and $d(\underset{\sim}{x}_j, \underset{\sim}{p})$ $(j = 1,\ldots,r)$ and these can be efficiently updated as $\underset{\sim}{p}$ changes as described by Galil and Kiefer (1980). The computation of $d(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})$ $(i = 1,\ldots,r)$ in (7) should also follow these rules.

G-optimality remains computationally expensive, however, but the calculation of (7) can often be further reduced by employing the bounds

$$0 \leq d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p}) \leq d(\underset{\sim}{x}_i, \underset{\sim}{p})\, d(\underset{\sim}{x}_j, \underset{\sim}{p}) \tag{9}$$

where the second inequality is that between a squared covariance and

variances. Therefore, when adding a point we have

$$d(\underset{\sim}{x}_i, \underset{\sim}{p}) - \frac{d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})}{d(\underset{\sim}{x}_j, \underset{\sim}{p})+n} \leq d(\underset{\sim}{x}_i, \underset{\sim}{p}). \tag{10}$$

When subtracting a point at $\underset{\sim}{x}_j$ we must have $p_j \geq n^{-1}$ and Theorem 1 of Atwood (1973) implies that $d(\underset{\sim}{x}_j, \underset{\sim}{p}) \leq n$; hence, the adjustment to all variances in (7) is nonnegative. Thus, when subtracting a point, the second inequality of (9) leads to

$$d(\underset{\sim}{x}_i, \underset{\sim}{p}) - \frac{d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p})}{d(\underset{\sim}{x}_j, \underset{\sim}{p}) - n} \leq \frac{d(\underset{\sim}{x}_i, \underset{\sim}{p})}{1-n^{-1}d(\underset{\sim}{x}_j, \underset{\sim}{p})} \tag{11}$$

The algorithm for the G-optimality maximization in (7) can be described in five steps.

S1. Sort $\mathcal{X}$ into the order $\underset{\sim}{x}_{<1>}, \ldots, \underset{\sim}{x}_{<r>}$ such that

$$d(\underset{\sim}{x}_{<1>}, \underset{\sim}{p}) \geq \ldots \geq d(\underset{\sim}{x}_{<r>}, \underset{\sim}{p}).$$

S2. Set MAX = 0. At completion MAX will represent the value of the maximum in (7). Now perform steps S3, S4 and S5 for i = 1, ..., r.

S3. For candidate $\underset{\sim}{x}_{<i>}$ compute the bound $d(\underset{\sim}{x}_{<i>}, \underset{\sim}{p})$ or $d(\underset{\sim}{x}_{<i>}, \underset{\sim}{p}) \left\{1 - n^{-1}d(\underset{\sim}{x}_j, \underset{\sim}{p})\right\}^{-1}$ in (10) or (11) as appropriate and call this $b_{<i>}$.

S4. If $b_{<i>} \leq$ MAX then

$$d(\underset{\sim}{x}_{<i>}, \underset{\sim}{p}) - \frac{d^2(\underset{\sim}{x}_{<i>}, \underset{\sim}{x}_j, \underset{\sim}{p})}{d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n} \leq \text{MAX}$$

and $\underset{\sim}{x}_{<i>}$ cannot increase the current value of MAX. Moreover, as the variances $d(\underset{\sim}{x}_{<i>}, \underset{\sim}{p})$ and the bounds $b_{<i>}$ are nonincreasing in i, neither can any $\underset{\sim}{x}_{<i'>}$ for $i' > i$ and the maximization algorithm terminates with the current value of MAX. If, however, $b_{<i>} >$ MAX, then continue to S5.

S5. Compute $d(\underset{\sim}{x}_{<i>}, \underset{\sim}{p}) - d^2(\underset{\sim}{x}_{<i>}, \underset{\sim}{x}_j, \underset{\sim}{p}) \left\{ d(\underset{\sim}{x}_j, \underset{\sim}{p}) \pm n \right\}^{-1}$

and increase MAX if necessary. Continue at S3 for the next i.

The implementation of these steps reduced execution time by approximately 60% in the example discussed in Section 3. It is also possible to utilize the second inequality in (9) in a similar way when computing the minimum in (3) for both G- and V-optimality but empirical studies showed the change to be ineffective.

For V-optimality, the calculation of $c(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_j)$ in (8) is facilitated by writing

$$\sum_{i=1}^{r} d^2(\underset{\sim}{x}_i, \underset{\sim}{x}_j, \underset{\sim}{p}) = \underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{M}(\underset{\sim}{p})^{-1} \sum_{i=1}^{r} \underset{\sim}{f}(\underset{\sim}{x}_i)\underset{\sim}{f}(\underset{\sim}{x}_i)^T \underset{\sim}{M}(\underset{\sim}{p})^{-1}\underset{\sim}{f}(\underset{\sim}{x}_j)$$

Moreover, these quantities need only be explicitly calculated for the initial design. Thereafter, as the current design changes, they and the associated matrix $\underset{\sim}{A}(\underset{\sim}{p}) = \underset{\sim}{M}(\underset{\sim}{p})^{-1} \Sigma \underset{\sim}{f}(\underset{\sim}{x}_i)\underset{\sim}{f}(\underset{\sim}{x}_i)^T \underset{\sim}{M}(\underset{\sim}{p})^{-1}$ can be updated using rules similar to those for $d(\underset{\sim}{x}_j, \underset{\sim}{p})$ $(j = 1, \ldots r)$ and $\underset{\sim}{M}(\underset{\sim}{p})^{-1}$ described by Galil and Kiefer (1980). When the design $\underset{\sim}{p}$ is replaced by $\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_{j_o}$, if we compute the two $k \times 1$ vectors $\underset{\sim}{a}_o = \underset{\sim}{A}(\underset{\sim}{p})\underset{\sim}{f}(\underset{\sim}{x}_{j_o})$ and $\underset{\sim}{m}_o = \underset{\sim}{M}(\underset{\sim}{p})^{-1}\underset{\sim}{f}(\underset{\sim}{x}_{j_o})$, then from (6) $\underset{\sim}{A}(\underset{\sim}{p})$ can be updated by

$$\underset{\sim}{A}(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_{j_o}) = \underset{\sim}{A}(\underset{\sim}{p}) - \frac{\underset{\sim}{a}_o\underset{\sim}{m}_o^T + \underset{\sim}{m}_o\underset{\sim}{a}_o^T}{d(\underset{\sim}{x}_{j_o}, \underset{\sim}{p}) \pm n} + \frac{\underset{\sim}{f}(\underset{\sim}{x}_{j_o})^T \underset{\sim}{A}(\underset{\sim}{p}) \underset{\sim}{f}(\underset{\sim}{x}_{j_o}) \underset{\sim}{m}_o\underset{\sim}{m}_o^T}{[d(\underset{\sim}{x}_{j_o}, \underset{\sim}{p}) \pm n]^2}$$

With this result, it is seen that $\underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{A}(\underset{\sim}{p}) \underset{\sim}{f}(\underset{\sim}{x}_j)$ $(j = 1, \ldots r)$ may be modified using the rule

$$\underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{A}(\underset{\sim}{p} \pm n^{-1}\underset{\sim}{\delta}_{j_o}) \underset{\sim}{f}(\underset{\sim}{x}_j) = \underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{A}(\underset{\sim}{p}) \underset{\sim}{f}(\underset{\sim}{x}_j) - \frac{2\underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{a}_o \underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{m}_o}{d(\underset{\sim}{x}_{j_o}, \underset{\sim}{p}) \pm n}$$

$$+ \frac{\underset{\sim}{f}(\underset{\sim}{x}_{j_o})^T \underset{\sim}{A}(\underset{\sim}{p}) \underset{\sim}{f}(\underset{\sim}{x}_{j_o}) [\underset{\sim}{f}(\underset{\sim}{x}_j)^T \underset{\sim}{m}_o]^2}{[d(\underset{\sim}{x}_{j_o}, \underset{\sim}{p}) \pm n]^2}$$

## 3. AN EXAMPLE

This example concerns a nonstandard design region of two explanatory variables $\underset{\sim}{x} = (x_1, x_2)$ defined by $0 \leq x_i \leq 0.7$ $(i = 1, 2)$ and $x_1 + x_2 \leq 1$. For our purposes the region is approximated by a grid of $r = 54$ points, where $x_i$ only takes values $0, 0.1, 0.2, \ldots, 0.7$ $(i = 1, 2)$. A second order model with $k = 6$ parameters

$$E(Y_{(i)}) = \beta_o + \beta_1 x_{(i)1} + \beta_2 x_{(i)2}$$

$$+ \beta_{11} x^2_{(i)1} + \beta_{22} x^2_{(i)2} + \beta_{12} x_{(i)1} x_{(i)2}$$

is to be fitted to the $n=6$ observations $\underset{\sim}{x}_{(i)} = (x_{(i)1}, x_{(i)2})$ $(i = 1, \ldots, 6)$.

For each of the design criteria D-, G- and V-optimality the generalized excursion algorithm was executed from ten random starts, resulting in the designs and properties listed in Table 1. For D- and V-optimality the ten attempts converged, for each criterion, to the same design subject to the interchange of $x_1$ and $x_2$. G-optimality gave less consistent results: only one try led to the displayed design.

If the objective of the experiment is response surface estimation, the D-optimal design performs badly here. The maximum variance $d_{max}(\underset{\sim}{p})$ is approximately 29% greater than that of the design produced by G-optimality and the average variance $d_{ave}(\underset{\sim}{p})$ about 8% larger than the V-optimal design. In fact, the V-optimal design performs well as measured by $d_{max}(\underset{\sim}{p})$ and $d_{ave}(\underset{\sim}{p})$ and would be recommended.

It can also be seen in Table 1 that the G- and V-optimality criteria are about 11 and 3 times more expensive than D-optimality. An examination of the computer operations involved suggests that for $r >> n, k$, if the number of times a point is added remains constant, these multiplicative factors are, in general, approximately $1 + (1-c)r/2$ and 2, where c reflects the time saved by the maximization algorithm in Section 2.4 (about 0.6 for the example).

## 4. CONCLUSIONS

The first objective of this study was to determine the magnitude of the possible improvements as measured by G- and V-optimality relative to the D-optimal design that can be found with the generalized

excursion algorithm. The example of Section 3 demonstrates that worthwhile decreases in the variance of the response surface estimator can, indeed, be effected. A second but important practical question concerns the time penalty relative to the computationally tractable D-optimality criterion. For V-optimality, the execution time is likely to be about twice that of D-optimality. The situation for G-optimality is less clear because of the problem-dependent factor c for time saved. For large r ($r > 100$ say) running time may be prohibitive and V-optimality would be more feasible.

A FORTRAN listing incorporating the above ideas is available from the author, but owners of an implemented DETMAX routine should find the necessary alterations straightforward.

## REFERENCES

ATWOOD, C. L. (1973). Sequences converging to D-optimal designs of experiments. Ann. Statist., 1, 342-352.

BOX, G. E. P. and DRAPER, N. R. (1959). A basis for the selection of a response surface design. J. Amer. Statist. Assoc., 54, 622-654.

GALIL, Z. and KIEFER, J. (1977). Comparison of Box-Draper and D-optimum designs for experiments with mixtures. Technometrics, 19, 441-444.

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.

KIEFER, J. (1975). Optimal design: variation in structure and performance under change of criterion. Biometrika, 62, 277-288.

KIEFER, J. and WOLFOWITZ, J. (1960). The equivalence of two extremum problems. Can. J. Math., 12, 363-366.

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

WELCH, W. J. (1981). Branch and bound search for experimental designs based upon D-optimality and other criteria. Technometrics, to appear.

Table 1. Designs and design properties (best of 10 attempts).

| Criterion | Design points | | | $\det\{M(\underset{\sim}{p})^{-1}\}$ | $d_{max}(\underset{\sim}{p})$ | $d_{ave}(\underset{\sim}{p})$ | Time/attempt (Cyber 174 secs) |
|---|---|---|---|---|---|---|---|
| D-optimality | (0,0) | (0,0.7) | (0.3,0.3) | .861E+10 | 10.38 | 5.57 | 0.7 |
| | (0.3,0.7) | (0.7,0) | (0.7,0.3) | | | | |
| G-optimality | (0,0) | (0,0.6) | (0.3,0.7) | .984E+10 | 8.07 | 5.38 | 7.6 |
| | (0.4,0.2) | (0.7,0) | (0.7,0.3) | | | | |
| V-optimality | (0,0) | (0,0.7) | (0.2,0.3) | .907E+10 | 8.07 | 5.16 | 2.2 |
| | (0.3,0.7) | (0.6,0) | (0.7,0.3) | | | | |

# PART 3

# A MEAN SQUARE ERROR CRITERION FOR THE DESIGN OF EXPERIMENTS

# 1. INTRODUCTION

In the design of experiments for response surface estimation the aim is to estimate a response $y(\underset{\sim}{x})$ over a specified region $\mathcal{H}$ of q explanatory variables $\underset{\sim}{x} = (x_1,\ldots,x_q)^T$. The design problem is to choose n not necessarily distinct points $\underset{\sim}{x}_{(i)}$ ( $i = 1,\ldots,n$) from $\mathcal{H}$ at which observations $Y(\underset{\sim}{x}_{(i)})$ are to be taken. It is usual to assume a linear model

$$E[Y(\underset{\sim}{x}_{(i)})] = y(\underset{\sim}{x}_{(i)}) = \underset{\sim}{f}^T(\underset{\sim}{x}_{(i)})\ \underset{\sim}{\beta} \qquad i = 1,\ldots,n \qquad (1)$$

where $\underset{\sim}{f}$ is a k x 1 vector of specified linearly independent functions on $\mathcal{H}$ and $\underset{\sim}{\beta}$ is a k x 1 vector of unknown parameters ($k \leq n$). It is often further assumed that the observations are uncorrelated with constant variance $\sigma^2$.

If the model (1) is correct, so capturing all of the systematic variation over $\mathcal{H}$, then design criteria such as the well-known D-optimality and G-optimality naturally consider only the sampling variance (for a review see Fedorov, 1972 or Silvey, 1980). In practice, however, the model (1) is always inadequate to some extent and, if least squares is used to estimate $\underset{\sim}{\beta}$ by $\underset{\sim}{\hat{\beta}}$, the estimated response $\hat{y}(\underset{\sim}{x}) = \underset{\sim}{f}^T(\underset{\sim}{x})\ \underset{\sim}{\hat{\beta}}$ $(\underset{\sim}{x} \in \mathcal{H})$ will differ from the true response $y(\underset{\sim}{x})$ due to model mis-specification as well as sampling variance.

Box and Draper (1959) proposed a primary criterion whereby the design is chosen to minimize the normalized integrated mean square error of $\hat{y}(\underset{\sim}{x})$ over $\mathcal{H}$, called J. As we shall only consider $\mathcal{H}$ comprising a finite set of

r candidate points $\{\underset{\sim}{x}_1,\ldots,\underset{\sim}{x}_r\}$ we can write J as

$$J = \frac{n}{r\sigma^2} \sum_{j=1}^{r} E[\hat{y}(\underset{\sim}{x}_j) - y(\underset{\sim}{x}_j)]^2$$

A design minimizing J shall be called J-optimal. Note that while Box and Draper integrate over the region $\mathcal{H}$ they allow the design points to be chosen from a larger operability region.

We can express J as the sum of two components: the normalized average variance

$$V = \frac{n}{r\sigma^2} \sum_{j=1}^{r} \text{Var}\, \hat{y}(\underset{\sim}{x}_j)$$

and the normalized average squared bias

$$B = \frac{n}{r\sigma^2} \sum_{j=1}^{r} [E\hat{y}(\underset{\sim}{x}_j) - y(\underset{\sim}{x}_j)]^2$$

The important point is that whereas V depends only on the design points, B (and hence J) also depends on $\sigma^2$ and the true responses $y(\underset{\sim}{x}_j)$ $(j = 1,\ldots,r)$ we are attempting to estimate. Box and Draper considered minimizing J when estimating a first-order model and the true response is, in fact, second-order. Remarkably, they found if $V \leq 4B$ the J-optimal design may be adequately approximated by the optimal all-bias design that completely ignores V. Moreover, optimal all-bias designs are readily constructed simply by matching the moments of the design points up to degree 3 with a design uniform over $\mathcal{H}$.

At least two problems remain, however, with this

pioneering approach. Galil and Kiefer (1977) compared all-bias and D-optimal designs for quadratic regressions when the true relationship also has one third degree parameter $\beta_{112}$ say, associated with $x_1^2 x_2$. They found that even if $\beta_{112}$ is as large as $\sigma$, the D-optimal design performed better as measured by J unless the sample size is fairly large. Secondly, the model assumption is only transferred one stage: the assumed true relationship, albeit more flexible than the fitted model, will itself always be inadequate to some extent.

Minimum bias estimation (Karson, Manson and Hader, 1969) replaces least squares estimation of the assumed polynomial model by an estimator minimizing B for any fixed design and parameter values in a specified higher-degree polynomial true response. Subject to this, the estimator also minimizes V for any fixed design and the remaining design flexibility is utilized in further reduction of V. The Kupper and Meydrech (1974), estimation-design approach requires the experimenter to furnish bounds on the parameters in the true relationship. In both of these methods, however, a "true" model still needs to be specified with the attendant practical difficulties.

Stigler (1971) suggests that the emphasis on bias stems partly from the use of an operability region O larger than $\mathcal{H}$. It O coincides with $\mathcal{H}$, as supposed in this paper, the tendency of variance-based designs to expand to the limits of O is constrained and their performance improved as measured by J.

## 2. A NEW MEAN SQUARE ERROR CRITERION

The design criterion described below attempts to protect against a larger class of alternatives to the assumed fitted model than permitted by a parametric generalization. In this sense the motivation is similar to that of Huber (1975). Estimation of the parameters in the fitted model will be by least squares throughout.

Without any loss of generality we may write the true responses as

$$y(\underset{\sim}{x}_j) = \underset{\sim}{f}^T(\underset{\sim}{x}_j)\underset{\sim}{\beta} + z_j \qquad j = 1,\ldots,r \tag{2}$$

with $z_j$ allowing for any discrepancy between the true response and the model (1) and let $z_{max} = \max |z_j|$ be an upper bound on the absolute departure. The experimenter will rarely be able to supply an estimate of $z_{max}$ but designs robust to this quantity can be constructed as shown in Section 3. To protect against the worst departures $\underset{\sim}{z} = (z_1,\ldots,z_r)^T$ we could choose the design points to minimize

$$J^* = V + \max_{\underset{\sim}{z}} B \tag{3}$$

and we now consider the behaviour of $J^*$ for any fixed design.

A design allocating $n_j$ observations to $\underset{\sim}{x}_j (j = 1,\ldots,r; \Sigma n_j = n)$ may be represented by a vector $\underset{\sim}{p} = (p_1,\ldots,p_r)^T$ where $p_j = n_j n^{-1}$. Let $\underset{\sim}{F}$ be the $r \times k$ matrix with row $j$ containing $\underset{\sim}{f}^T(\underset{\sim}{x}_j)$ and $\underset{\sim}{P}$ the $r \times r$ diagonal matrix $\text{diag}(\underset{\sim}{p})$. A normalized version of the familiar $\underset{\sim}{X}^T\underset{\sim}{X}$ matrix can be defined as

$$\underset{\sim}{M}(p) = \underset{\sim}{F}^T \underset{\sim}{P}\, \underset{\sim}{F}$$

whereupon, if $\underset{\sim}{M}(p)$ is nonsingular, the variance component of J can be written

$$V(p) = \frac{1}{r} \sum \underset{\sim}{f}^T(x_j)\, \underset{\sim}{M}^{-1}(p)\, \underset{\sim}{f}(x_j)$$

$$= \text{trace}\ \underset{\sim}{M}^{-1}(p)\, \underset{\sim}{M}(p_{unif})$$

and $p_{unif}$ is the uniform design $p_1 = \ldots = p_r = r^{-1}$.

The r x 1 vector of estimated responses $\hat{y} = F \hat{\beta}$ has expectation

$$E(\hat{y}) = F\, M^{-1}(p)\, F^T P\, y$$

$$= F\, \beta + F\, M^{-1}(p)\, F^T P z$$

from (2). If for notational convenience we write the k x r matrix $M^{-1}(p)\, F^T P$ as $Q(p)$, then the bias component of J is a function of the departures $z$ given by

$$B(p,z) = \frac{n}{r\sigma^2} z^T [F\, Q(p) - I]^T [F\, Q(p) - I] z \qquad (4)$$

For any fixed design $p$, $B(p,z)$ is seen to be a non-negative definite quadratic form in $z$ and hence convex in $z$; that is, for any two departures $z_1$ and $z_2$

$$B[\underset{\sim}{p},\alpha\underset{\sim}{z}_1 + (1-\alpha)\underset{\sim}{z}_2] \leq \alpha B(\underset{\sim}{p},\underset{\sim}{z}_1) + (1-\alpha)\, B(\underset{\sim}{p},\underset{\sim}{z}_2) \quad (0 \leq \alpha \leq 1)$$

Therefore, for any fixed design $\underset{\sim}{p}$, maximum $B(\underset{\sim}{p},\underset{\sim}{z})$ over $\underset{\sim}{z}$ subject to $|z_j| \leq z_{max}$ $(j = 1,\ldots,r)$ is always achieved at one of the $2^r$ extremes $\underset{\sim}{z} = (\pm z_{max},\ldots,\pm z_{max})^T$.

Although the maximization of B in (3) is therefore confined to a finite set of $\underset{\sim}{z}$'s, J* will nevertheless be difficult to compute when r is large. A second unappealing property of J* is a discontinuous first derivative with respect to $\underset{\sim}{p}$, leading to problems in numerically optimizing the design. A more tractable criterion is to choose the design $\underset{\sim}{p}$ to minimize

$$J^{**}(\underset{\sim}{p}) = V(\underset{\sim}{p}) + 2^{-r}\Sigma\, B(\underset{\sim}{p},\underset{\sim}{z})$$

with summation over the $2^r$ extremes $\underset{\sim}{z} = (\pm z_{max},\ldots,\pm z_{max})^T$. Hence, J** averages J over the $2^r$ potential worst-case departures from the assumed model. This criterion possesses a continuous first derivative and, furthermore, the average over the $2^r$ extremes does not require explicit computation of the $2^r$ components since, from (4),

$$2^{-r}\,\Sigma\, B(\underset{\sim}{p},\underset{\sim}{z}) = \frac{2^{-r}n}{r\sigma^2}\ \text{trace}\ [\underset{\sim\sim}{F}\underset{\sim}{Q}(\underset{\sim}{p})-\underset{\sim}{I}]^T[\underset{\sim\sim}{F}\underset{\sim}{Q}(\underset{\sim}{p})-\underset{\sim}{I}]\Sigma\underset{\sim\sim}{z}\underset{\sim\sim}{z}^T$$

and $2^{-r}\Sigma\underset{\sim\sim}{z}\underset{\sim\sim}{z}^T = z_{max}^2\,\underset{\sim}{I}$. Further simplication follows by noting that $\text{trace}\ \underset{\sim\sim}{F}\underset{\sim}{Q}(\underset{\sim}{p}) = k$ whereupon $J^{**}(\underset{\sim}{p})$ can be written as

$$J^{**}(\underset{\sim}{p}) = V(\underset{\sim}{p}) + nz_{max}^2\sigma^{-2}\, B^{**}(\underset{\sim}{p}) \qquad (5)$$

where

$$B^{**}(\underset{\sim}{p}) = \text{trace } \underset{\sim}{Q}(\underset{\sim}{p})\underset{\sim}{Q}^T(\underset{\sim}{p})\, \underset{\sim}{M}(\underset{\sim}{p}_{unif}) + 1 - 2kr^{-1} \quad .$$

and $nz_{max}^2\sigma^{-2}$ determines the relative importance of bias in the criterion.

## 3. DESIGN CONSTRUCTION

In this section two algorithms for computer-construction of J**-optimal designs are outlined. Both have been implemented in FORTRAN and listings are available from the author.

### 3.1 Construction of approximate designs.

It is convenient to first consider the computation of optimal approximate designs where $p_j$ need not be an integer multiple of $n^{-1}$ $(j = 1,\ldots,r)$. In practice, the best exact design with integer replications will be required, but the optimal approximate design may offer a computationally inexpensive guide to exact design selection. The optimization problem may be written as minimize $J^{**}(\underset{\sim}{p})$ defined in (5) subject to

$$\ell_j \leq p_j \leq u_j \qquad j = 1,\ldots,r \tag{6}$$

and

$$\Sigma p_j = 1 \tag{7}$$

where the lower and upper bounds $\ell_j$ and $u_j$ on $p_j$ are specified constants $(0 \leq \ell_j \leq u_j \leq 1)$. In most cases we would set $\ell_j = 0$, $u_j = 1$ $(j = 1,\ldots,r)$ but the more general formulation allows, for instance, design augmentation where some of the $p_j$ have positive lower bounds as a number of observations have already been taken.

The algorithm proposed here to minimize $J^{**}(\underset{\sim}{p})$ subject to (6) and (7) is similar to that of Fedorov (1972, pp. 97-104) for constructing D-optimal designs. Apart from the

obvious change in criterion, the major difference is that two elements of the design $\underset{\sim}{p}$ are altered at each iteration rather than just one to facilitate the constraints (6). The algorithm can be described in five steps.

S1. Choose an arbitrary starting design $\underset{\sim}{p}^{(0)}$ satisfying (6) and (7) such that $\underset{\sim}{M}(\underset{\sim}{p}^{(0)})$ is nonsingular.

S2. Calculate the r gradients

$$g_j(\underset{\sim}{p}^{(0)}) = \frac{d}{d\alpha} J^{**}(\underset{\sim}{p}^{(0)} + \alpha \underset{\sim}{\delta}_j)\Big|_{\alpha = 0} \quad j = 1,\ldots,r \qquad (8)$$

from $\underset{\sim}{p}^{(0)}$ in the direction of the designs $\underset{\sim}{\delta}_j$ with unity in position j and zero elsewhere.

S3. Find h and i satisfying

$$g_h(\underset{\sim}{p}^{(0)}) = \min_{\substack{j = 1,\ldots,r \\ p_j < u_j}} g_j(\underset{\sim}{p}^{(0)})$$

and

$$g_i(\underset{\sim}{p}^{(0)}) = \max_{\substack{j = 1,\ldots,r \\ p_j > \ell_j}} g_j(\underset{\sim}{p}^{(0)})$$

S4. Form a new design $\underset{\sim}{p}^{(1)} = \underset{\sim}{p}^{(0)} + \alpha\,(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)$ corresponding to a transfer of $\alpha$ from $p_i^{(0)}$ to $p_h^{(0)}$ with $\alpha$ chosen to minimize $J^{**}(\underset{\sim}{p}^{(1)})$ subject to $p_i^{(0)} - \ell_i \leq \alpha \leq u_h - p_h^{(0)}$, a restriction ensuring that $\underset{\sim}{p}^{(1)}$ satisfies (6). The condition (7) is automatically satisfied. We always have $J^{**}(\underset{\sim}{p}^{(1)}) \leq J^{**}(\underset{\sim}{p}^{(0)})$ as equality is achieved simply

by setting $\alpha = 0$.

S5. Repeat steps S2, S3 and S4 replacing $\underset{\sim}{p}^{(0)}$ by $\underset{\sim}{p}^{(1)}$ and continue iterating until the sequence $\underset{\sim}{p}^{(s)}$ $(s = 0,1,2,\ldots)$ with $J(\underset{\sim}{p}^{(s+1)}) \leq J(\underset{\sim}{p}^{(s)})$ is sufficiently close to an optimal design.

The details of computing $g_j(\underset{\sim}{p})$ and $\alpha$ in steps S2 and S4 are given in Appendix 1.

Empirical studies not reported here indicate that the algorithm always converges to a unique value of J** irrespective of the initial design: it would therefore appear that a global optimum is provided. The sequence $\underset{\sim}{p}^{(s)}$ $(s = 0,1,2,\ldots)$ is terminated when $J^{**}(\underset{\sim}{p}^{(s)}) \leq (1 + \tau)\, J^{**}_{1b}(\underset{\sim}{p}^{(s)})$ where $\tau$ is a suitable tolerance (say 0.001) and $J^{**}_{1b}(\underset{\sim}{p}^{(s)})$ is an approximate lower bound on the limiting value of J** computed as

$$J^{**}_{1b}(\underset{\sim}{p}^{(s)}) = J^{**}(\underset{\sim}{p}^{(s)}) + \min_{\underset{\sim}{p}' \text{ satisfying (6) and (7)}} \sum_{j=1}^{r} p'_j [V(\underset{\sim}{p}^{(s)}) + g_j(\underset{\sim}{p}^{(s)})]$$

This bound may be derived by assuming that J** is approximately convex along linear combinations $(1 - \alpha)\underset{\sim}{p}^{(s)} + \alpha \underset{\sim}{p}'$ between $\underset{\sim}{p}^{(s)}$ and an (unknown) optimal design $\underset{\sim}{p}'$ also satisfying (6) and (7).

### 3.2 An example (approximate designs)

Suppose there are two explanatory variables with the design region $\mathcal{X}$ comprising the r = 9 combinations of each variable at levels -1, 0 or 1. The assumed model is $EY_{(i)} = \beta_0 + \beta_1 x_{(i)1} + \beta_2 x_{(i)2}$ $(i = 1,\ldots,n)$. The J**

-optimal designs and their properties are displayed in Table 1 for $nz^2_{max}\sigma^{-2}$ taking values of 0,2,8 and infinity. Because of symmetries in the design region only three elements of $\underset{\sim}{p}$ need be reported: $p_1$ is the proportion of the observations at each of the four corner points $(\pm1,\pm1)$; $p_2$ the proportion at the four mid-edges $(\pm1,0)$ and $(0,\pm1)$, and $p_3$ the proportion at the centre (0,0). At $nz^2_{max}\sigma^{-2}= 0$ when only variance is considered all of the observations are taken at the corners, but as $nz^2_{max}\sigma^{-2}$ is increased and bias becomes more important the optimal design tends to the uniform design with a corresponding increase in V and decrease in B**. By an argument similar to that of Box and Draper (1963, Appendix 1) it may be demonstrated that a uniform design always minimizes $B^{**}(\underset{\sim}{p})$.

We may also use this example to illustrate how the experimenter may make a robust choice of $nz^2_{max}\sigma^{-2}$ if unable to supply a value. The percent efficiency of a design $\underset{\sim}{p}$ relative to the J**-optimal design $\underset{\sim}{p}^{(opt)}$ is a function of $nz^2_{max}\sigma^{-2}$

$$e(\underset{\sim}{p},nz^2_{max}\sigma^{-2}) = 100\ J^{**}(\underset{\sim}{p}^{(opt)})\ [J^{**}(\underset{\sim}{p})]^{-1}$$

and for $\underset{\sim}{p}$ to be robust we require that the minimum efficiency over the range $0 \leq nz^2_{max}\sigma^{-2} < \infty$ is large. Furthermore, it may be shown that this minimum is either $e(\underset{\sim}{p},0)$ or the limit $e(\underset{\sim}{p},\infty)$: hence, the most severe tests of the robustness of $\underset{\sim}{p}$ are the comparisons with the optimal all-variance and all-bias designs. In Table 1 $e(\underset{\sim}{p},0)$ and $e(\underset{\sim}{p},\infty)$ are supplied

for the four computed designs and it is seen that the second, constructed by setting $nz_{max}^2\sigma^{-2} = 2$, is the most robust in the above sense with a minimum efficiency of nearly 91%. Thus, the penalty for not specifying $nz_{max}^2\sigma^{-2}$ is remarkably small in this case.

## 3.3 Construction of exact designs

When we impose the condition that $p_j (j = 1,\ldots,r)$ is an integer multiple of $n^{-1}$, the construction of J**-optimal designs encounters considerable computational obstacles. Welch (1981) shows that the DETMAX algorithm (Mitchell, 1974) may be adapted to a general criterion minimize $C(\underset{\sim}{p})$ subject to (6), (7) and the integer replication constraint

$$p_j = n_j n^{-1} \qquad n_j \text{ an integer} \qquad j = 1,\ldots,r \qquad (9)$$

In our case $C(\underset{\sim}{p}) = J^{**}(\underset{\sim}{p})$. This algorithm may, however, generate a local optimum.

Following DETMAX, the generalized algorithm commences with an initial random design of n points satisfying (6), (7) and (9) and makes a number of excursions whereby the design is improved until no further progress can be made. Each excursion is a series of additions or subtractions of a design point to the current collection of n' points $\underset{\sim}{p}_{n'}$, eventually returning to a possibly new and better n-point design. The criterion J** affects the algorithm only in the choice of a promising point to add or subtract as required. If a design point $\underset{\sim}{x}_{jo}$ is added or subtracted, $\underset{\sim}{p}_{n'}$ is updated to $\underset{\sim}{p}_{n'} \pm n^{-1}\,\underset{\sim}{\delta}_j$. Accordingly, a reasonable choice is $\underset{\sim}{x}_{jo}$ minimizing

$$(n' \pm 1)n^{-1}\, V(p_{\sim n}, \pm n^{-1}\delta_{\sim j}) + nz_{max}^2\sigma^{-2}B^{**}(p_{\sim n}, \pm\, n^{-1}\, \delta_{\sim j}) \qquad (10)$$

where V is multiplied by $(n' \pm 1)n^{-1}$ to normalize the variance component as we shall have $\Sigma p_j = (n' \pm 1)n^{-1}$ after the change. The computation of (10) is described in Appendix 1.

3.4 An example (exact designs)

When the excursion algorithm for exact design construction is applied to the example described in section 3.2, with n = 8 observations, only the two designs displayed in Table 2 can result. At $z_{max}\sigma^{-1} = 0$ the optimal all-variance exact design coincides with the approximate V-optimal design, since the latter is also exact for n = 8. Furthermore, this design remains J**-optimal when bias is included for $z_{max}\sigma^{-1}$ less than approximately 0.49; any value of $z_{max}\sigma^{-1}$ above this range results in a design concurring with the all-bias optimal exact design ignoring V completely. The dichotomy seen here is a reflection of the simplicity of the example; more complex situations will often produce a less abrupt progression of designs. In this illustration, however, the choice of design is determined by the experimenter's belief that the maximum deviation of the true response from the assumed model is either less than or greater than about half the sampling error standard deviation. If a robust choice in the sense of section 3.2 is to be made, the last two columns of Table 2 indicate that the all-bias design, with a minimum efficiency of about 84%, is to be preferred. With a

different number of observations the optimal all-bias design would not necessarily be the robust selection. For example, when $n = 9$ a design intermediate between the all-bias and all-variance designs is indicated.

# 4. CONCLUSIONS

Any experimental design criterion that admits the possibility of bias arising from an inadequate model specification must surmount two obstacles. Firstly, the nature of the possible alternatives to the assumed model has to be defined and, secondly, the importance of the bias arising from such alternatives, relative to the error variance, needs to be known. However, in most practical situations, little or no information pertaining to these two questions will be available. The approach described here attempts to proceed in a robust manner at both stages, by protecting against a wide class of non-parametric alternatives and, then, choosing a compromise between bias and variance with a large efficiency relative to any "correct" balance.

The two algorithms supplied also enable designs to be constructed for the adopted criterion. The approximate design algorithm will typically demand less computer time than for exact designs, an advantage more apparent for large problems. If the approximate algorithm cannot offer a reasonable guide to exact design selection, empirical studies have suggested that it may nonetheless be helpful in searching for a robust balance between variance and bias at which the exact procedure may be executed.

APPENDIX 1. Some computational details.

The choice of the computational procedures involved in implementing the algorithms of section 3 can have an important effect on execution time. By extending the time-saving operations that Galil and Kiefer (1980) employed to maintain the essential quantities for the D-optimality criterion, an efficient implementation can also be achieved for J**-optimality.

The approximate and exact design algorithms both start with an initial design for which we need to initialize certain variables. These quantities enable the first iteration to be executed; they can then be updated to facilitate the second and subsequent iterations. We commence by calculating the four k x k matrices $\underset{\sim}{M}^{-1}(\underset{\sim}{p})$, $\underset{\sim}{M}^{-1}(\underset{\sim}{p})\underset{\sim}{M}(\underset{\sim}{p}_{unif})\underset{\sim}{M}^{-1}(\underset{\sim}{p})$, $\underset{\sim}{Q}(\underset{\sim}{p})\underset{\sim}{Q}^{T}(\underset{\sim}{p})$ and $\underset{\sim}{G}(\underset{\sim}{p}) = [\underset{\sim}{H}(\underset{\sim}{p}) + \underset{\sim}{H}^{T}(\underset{\sim}{p})]/2$, with $\underset{\sim}{H}(\underset{\sim}{p})$ defined as $\underset{\sim}{Q}(\underset{\sim}{p})\underset{\sim}{Q}^{T}(\underset{\sim}{p})\underset{\sim}{M}(\underset{\sim}{p}_{unif})\underset{\sim}{M}^{-1}(\underset{\sim}{p})$, for the starting design $\underset{\sim}{p}$. With these, we can compute the quadratic forms $a_{jj}(\underset{\sim}{p})$, $b_{jj}(\underset{\sim}{p})$, $c_{jj}(\underset{\sim}{p})$ and, in the case of the excursion algorithm, $d_{jj}(\underset{\sim}{p})$ $(j = 1,\ldots,r)$ where

$$a_{st}(\underset{\sim}{p}) = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\ \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})\ \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_t) \tag{11}$$

$$b_{st}(\underset{\sim}{p}) = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\ \underset{\sim}{Q}(\underset{\sim}{p})\ \underset{\sim}{Q}^{T}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_t) \tag{12}$$

$$c_{st}(\underset{\sim}{p}) = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\ \underset{\sim}{G}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_t) \tag{13}$$

and

$$d_{st}(\underset{\sim}{p}) = \underset{\sim}{f}^T(\underset{\sim}{x}_s)\ \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_t) \quad . \tag{14}$$

The approximate algorithm will only require a few of the r values of $d_{jj}(\underset{\sim}{p})$; these can be computed as necessary. To initialize $J^{**}(\underset{\sim}{p})$, the traces of $\underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})$ and $\underset{\sim}{Q}(\underset{\sim}{p})\ \underset{\sim}{Q}^T(\underset{\sim}{p})\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})$ appearing in $\underset{\sim}{V}(\underset{\sim}{p})$ and $\underset{\sim}{B}^{**}(\underset{\sim}{p})$ also need to be calculated. With the above quadratic forms it is possible to compute the gradients in (8) or V and B** in (10) for the approximate and exact algorithms respectively by considering the adjusted design $\underset{\sim}{p} + \alpha\ \underset{\sim}{\delta}_j$.

From Theorem 2.6.1 of Fedorov (1972, pp. 106-107)

$$\underset{\sim}{M}^{-1}(\underset{\sim}{p} + \alpha\ \underset{\sim}{\delta}_j) = \underset{\sim}{M}^{-1}(\underset{\sim}{p}) - [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\underset{\sim}{d}_{\cdot j}(\underset{\sim}{p})\underset{\sim}{d}_{\cdot j}^T(\underset{\sim}{p}) \tag{15}$$

where $\underset{\sim}{d}_{\cdot j}(\underset{\sim}{p})$ is the k x 1 vector

$$\underset{\sim}{d}_{\cdot j}(\underset{\sim}{p}) = \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_j) \tag{16}$$

and we may obtain $\underset{\sim}{V}(\underset{\sim}{p} + \alpha\ \underset{\sim}{\delta}_j)$ by postmultiplying (15) by $\underset{\sim}{M}(\underset{\sim}{p}_{unif})$ and taking the trace to yield

$$\underset{\sim}{V}(\underset{\sim}{p} + \alpha\ \underset{\sim}{\delta}_j) = \underset{\sim}{V}(\underset{\sim}{p}) - [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\ a_{jj}(\underset{\sim}{p}) \tag{17}$$

To determine the corresponding adjustment for B** we first substitute (15) in $\underset{\sim}{M}^{-1}(\underset{\sim}{p} + \alpha\ \underset{\sim}{\delta}_j)\ [\underset{\sim}{F}^T\underset{\sim}{P} + \alpha\underset{\sim}{f}(\underset{\sim}{x}_j)\underset{\sim}{\delta}_j^T]$ to find

$$\underset{\sim}{Q}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = \underset{\sim}{Q}(\underset{\sim}{p}) - [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\underset{\sim}{d}_{\cdot j}(\underset{\sim}{p})\ [\underset{\sim}{f}^T(\underset{\sim}{x}_j)\underset{\sim}{Q}(\underset{\sim}{p}) - \underset{\sim}{\delta}_j^T] \quad .$$

It then follows that

$$\underset{\sim}{Q}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j)\ \underset{\sim}{Q}^T(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = \underset{\sim}{Q}(\underset{\sim}{p})\ \underset{\sim}{Q}^T(\underset{\sim}{p})$$

$$-[\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}[\underset{\sim}{b}_{\cdot j}(\underset{\sim}{p})\underset{\sim}{d}_{\cdot j}^T(\underset{\sim}{p}) + \underset{\sim}{d}_{\cdot j}(\underset{\sim}{p})\underset{\sim}{b}_{\cdot j}^T(\underset{\sim}{p})]$$

$$+[\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-2}[1 + 2p_j\alpha^{-1} + b_{jj}(\underset{\sim}{p})]\ \underset{\sim}{d}_{\cdot j}(\underset{\sim}{p})\underset{\sim}{d}_{\cdot j}^T(\underset{\sim}{p}) \qquad (18)$$

with the k x 1 vector $\underset{\sim}{b}_{\cdot j}(\underset{\sim}{p})$ defined by

$$\underset{\sim}{b}_{\cdot j}(\underset{\sim}{p}) = \underset{\sim}{Q}(\underset{\sim}{p})\ \underset{\sim}{Q}^T(\underset{\sim}{p})\ \underset{\sim}{f}(x_j) \qquad (19)$$

After postmultiplying (18) by $\underset{\sim}{M}(\underset{\sim}{p}_{unif})$ and taking the trace we obtain the desired result

$$B^{**}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = B^{**}(\underset{\sim}{p}) - 2[\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\ c_{jj}(\underset{\sim}{p})$$

$$+ [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-2}[1 + 2p_j\alpha^{-1} + b_{jj}(\underset{\sim}{p})]a_{jj}(\underset{\sim}{p}) \quad . \qquad (20)$$

The computation of (10) for the exact algorithm immediately follows from (17) and (20). Differentiation of (17) and (20) with respect to $\alpha$ and setting $\alpha = 0$ gives the gradients in (8) for the approximate algorithm as

$$g_j(\underset{\sim}{p}) = -\ a_{jj}(\underset{\sim}{p}) + 2nz_{max}^2\bar{\sigma}^{-2}[p_j a_{jj}(\underset{\sim}{p}) - c_{jj}(\underset{\sim}{p})] \quad .$$

To complete the first iteration of the approximate algorithm, there remains the problem of choosing $\alpha$ in

step s4 when the design $\underset{\sim}{p}$ is changed to $\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)$ with h and i fixed. The implemented method optimizes $\alpha$ numerically. The expressions for $V[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)]$ and $B^{**}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)]$ appearing below allow $J^{**}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)]$ to be numerically minimized as a function of $\alpha$ by any minimization routine only requiring function values, for example the quadratic interpolation method E04ABF available in the NAG library of subroutines (NAG, 1978).

We need to compute the quadratic forms $d_{hh}(\underset{\sim}{p})$ and $d_{ii}(\underset{\sim}{p})$ given in (14) that are uninitialized by the approximate algorithm and the further four quantities $a_{hi}(\underset{\sim}{p})$, $b_{hi}(\underset{\sim}{p})$, $c_{hi}(\underset{\sim}{p})$, and $d_{hi}(\underset{\sim}{p})$ defined in (11), (12), (13) and (14). To find the modified value of V, we first use the result (Lemma 3.3.1 of Fedorov, 1972, p. 167)

$$\underset{\sim}{M}^{-1}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)] = \underset{\sim}{M}^{-1}(\underset{\sim}{p}) - \underset{\sim}{M}^{-1}(\underset{\sim}{p})\, \underset{\sim}{F}_{hi}^{T}\, \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p})\, \underset{\sim}{F}_{hi}\, \underset{\sim}{M}^{-1}(\underset{\sim}{p}) \tag{21}$$

where $\underset{\sim}{F}_{hi}$ is the $2 \times k$ matrix with rows $\underset{\sim}{f}^{T}(\underset{\sim}{x}_h)$ and $\underset{\sim}{f}^{T}(\underset{\sim}{x}_i)$ and $\underset{\sim}{D}_{hi}(\underset{\sim}{p})$ is the $2 \times 2$ matrix

$$\underset{\sim}{D}_{hi}(\underset{\sim}{p}) = \begin{bmatrix} \alpha^{-1} + d_{hh}(\underset{\sim}{p}) & d_{hi}(\underset{\sim}{p}) \\ d_{hi}(\underset{\sim}{p}) & -\alpha^{-1} + d_{ii}(\underset{\sim}{p}) \end{bmatrix}.$$

By an argument similar to that leading to (17), the change in V follows from (21) and is given by

$$V[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)] = V(\underset{\sim}{p}) - \operatorname{trace} \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p})\, \underset{\sim}{A}_{hi}(\underset{\sim}{p}) \tag{22}$$

with $\underset{\sim}{A}_{hi}(\underset{\sim}{p})$ defined as the $2 \times 2$ matrix

$$\underset{\sim}{A}_{hi}(\underset{\sim}{p}) = \begin{bmatrix} a_{hh}(\underset{\sim}{p}) & a_{hi}(\underset{\sim}{p}) \\ a_{hi}(\underset{\sim}{p}) & a_{ii}(\underset{\sim}{p}) \end{bmatrix}$$

To compute the corresponding change in B**, we note that from (21)

$$\underset{\sim}{Q}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)] = \underset{\sim}{Q}(\underset{\sim}{p}) - \underset{\sim}{M}^{-1}(\underset{\sim}{p})\underset{\sim}{F}_{hi}^T \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p})[\underset{\sim}{F}_{hi}\underset{\sim}{Q}(\underset{\sim}{p}) - \underset{\sim}{\Delta}_{hi}^T]$$

where $\underset{\sim}{\Delta}_{hi}$ represents the r x 2 matrix $(\underset{\sim}{\delta}_h \quad \underset{\sim}{\delta}_i)$. The expression for B** then follows by substituting in trace $\underset{\sim}{Q}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)]\ \underset{\sim}{Q}^T[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)]\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})$ whereupon

$$B^{**}[\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)] = B^{**}(\underset{\sim}{p}) - 2 \text{ trace } \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p})\ \underset{\sim}{T}_{hi}(\underset{\sim}{p})$$

$$+ \text{ trace } \underset{\sim}{S}_{hi}(\underset{\sim}{p})\ \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p})\ \underset{\sim}{A}_{hi}(\underset{\sim}{p})\ \underset{\sim}{D}_{hi}^{-1}(\underset{\sim}{p}) \qquad (23)$$

with the further 2 x 2 matrices $\underset{\sim}{T}_{hi}(\underset{\sim}{p})$ and $\underset{\sim}{S}_{hi}(\underset{\sim}{p})$ defined as

$$\underset{\sim}{T}_{hi}(\underset{\sim}{p}) = \begin{bmatrix} c_{hh}(\underset{\sim}{p}) - p_h a_{hh}(\underset{\sim}{p}) & c_{hi}(\underset{\sim}{p}) - (p_h + p_i)a_{hi}(\underset{\sim}{p})/2 \\ c_{hi}(\underset{\sim}{p}) - (p_h + p_i)a_{hi}(\underset{\sim}{p})/2 & c_{ii}(\underset{\sim}{p}) - p_i\, a_{ii}(\underset{\sim}{p}) \end{bmatrix}$$

and

$$\underset{\sim}{S}_{hi}(\underset{\sim}{p}) = \begin{bmatrix} b_{hh}(\underset{\sim}{p}) - 2p_h d_{hh}(\underset{\sim}{p}) + 1 & b_{hi}(\underset{\sim}{p}) - (p_h + p_i)d_{hi}(\underset{\sim}{p}) \\ b_{hi}(\underset{\sim}{p}) - (p_h + p_i)\, d_{hi}(\underset{\sim}{p}) & b_{ii}(\underset{\sim}{p}) - 2p_i d_{ii}(\underset{\sim}{p}) + 1 \end{bmatrix}$$

Hence, with (22) and (23) the numerical optimization of $\alpha$ is computationally straightforward.

At the end of the first and subsequent iterations one element of the design $\underset{\sim}{p}$ is changed by the exact algorithm and two are altered by the approximate algorithm. To facilitate the next iteration,we now update the quantities that were initialized by considering the new values when $\underset{\sim}{p}$ is replaced by $\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j$ say. The approximate algorithm requires two applications of these results: $\underset{\sim}{p} + \alpha(\underset{\sim}{\delta}_h - \underset{\sim}{\delta}_i)$ can be written as $(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_h) - \alpha\underset{\sim}{\delta}_i$ and we update first with $j = h$ to give an intermediate design and then again with $j = i$.

We first compute the four $k \times 1$ vectors

$$\underset{\sim}{a}_{\cdot j} = \underset{\sim}{M}^{-1}(\underset{\sim}{p})\underset{\sim}{M}(\underset{\sim}{p}_{unif})\ \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_j), \qquad \underset{\sim}{b}_{\cdot j} \text{ in (19)},$$

$$\underset{\sim}{c}_{\cdot j} = \underset{\sim}{G}(\underset{\sim}{p})\ \underset{\sim}{f}(\underset{\sim}{x}_j), \qquad \text{and } \underset{\sim}{d}_{\cdot j} \text{ in (16)},$$

where the dependence of these vectors on the unmodified design $\underset{\sim}{p}$ is omitted for brevity. Similarly, as the vector $\underset{\sim}{d}_{\cdot j}$ will often appear with the scalar multiplier $[\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}$ it will be convenient to write $\underset{\sim}{\bar{d}}_{\cdot j} = [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\ \underset{\sim}{d}_{\cdot j}$ .

To update the initialized matrices, expressions for $\underset{\sim}{M}^{-1}$ and $\underset{\sim\sim}{QQ}^T$ are already available in (15) and (18). From these two results, the remaining two matrices can be adjusted by applying

$$\underset{\sim}{M}^{-1}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j)\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})\ \underset{\sim}{M}^{-1}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = \underset{\sim}{M}^{-1}(\underset{\sim}{p})\ \underset{\sim}{M}(\underset{\sim}{p}_{unif})\ \underset{\sim}{M}^{-1}(\underset{\sim}{p})$$

$$- \underset{\sim}{\bar{d}}_{\cdot j}\ \underset{\sim}{a}_{\cdot j}^{T} - \underset{\sim}{a}_{\cdot j}\ \underset{\sim}{\bar{d}}_{\cdot j}^{T} + a_{jj}\ \underset{\sim}{\bar{d}}_{\cdot j}\ \underset{\sim}{\bar{d}}_{\cdot j} \qquad (24)$$

and

$$\underset{\sim}{G}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = \underset{\sim}{G}(\underset{\sim}{p}) - \underset{\sim}{\bar{d}}_{\cdot j}\underset{\sim}{c}_{\cdot j}^{T} - \underset{\sim}{c}_{\cdot j}\underset{\sim}{\bar{d}}_{\cdot j}^{T} + c_{jj}(\underset{\sim}{p})\underset{\sim}{\bar{d}}_{\cdot j}\underset{\sim}{\bar{d}}_{\cdot j}^{T}$$

$$- [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}\ [\underset{\sim}{v}_{\cdot j}\ \underset{\sim}{w}_{\cdot j}^{T} + \underset{\sim}{w}_{\cdot j}\ \underset{\sim}{v}_{\cdot j}^{T}]/2 \qquad (25)$$

where $\underset{\sim}{v}_{\cdot j} = \underset{\sim}{b}_{\cdot j} - [1 + 2p_j\ \alpha^{-1} + b_{jj}(\underset{\sim}{p})]\ \underset{\sim}{\bar{d}}_{\cdot j}$ and

$$\underset{\sim}{w}_{\cdot j} = \underset{\sim}{a}_{\cdot j} - a_{jj}(\underset{\sim}{p})\ \underset{\sim}{\bar{d}}_{\cdot j} \quad .$$

The updating rules for the quadratic forms $a_{ss}$, $b_{ss}$, $c_{ss}$ and, in the case of the exact algorithm, $d_{ss}$ $(s = 1,\ldots,r)$ follow from the corresponding matrix operations (24), (18), (25) and (15). For $s = 1,\ldots,r$, once we have computed the five scalars $a_{sj} = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\ \underset{\sim}{a}_{\cdot j}$, $b_{sj} = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\underset{\sim}{b}_{\cdot j}$, $c_{sj} = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\ \underset{\sim}{c}_{\cdot j}$, $d_{sj} = \underset{\sim}{f}^{T}(\underset{\sim}{x}_s)\underset{\sim}{d}_{\cdot j}$ and $\bar{d}_{sj} = [\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}d_{sj}$, the quadratic forms can be reset by

$$a_{ss}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = a_{ss}(\underset{\sim}{p}) - 2a_{sj}\bar{d}_{sj} + a_{jj}(\underset{\sim}{p})\bar{d}_{sj}^{2}$$

$$b_{ss}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = b_{ss}(\underset{\sim}{p}) - 2b_{sj}\bar{d}_{sj} + [1 + 2p_j\alpha^{-1} + b_{jj}(\underset{\sim}{p})]\bar{d}_{sj}^{2}$$

$$c_{ss}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = c_{ss}(\underset{\sim}{p}) - 2c_{sj}\bar{d}_{sj} + c_{jj}(\underset{\sim}{p})\bar{d}_{sj}^{2}$$

$$-[\alpha^{-1} + d_{jj}(\underset{\sim}{p})]^{-1}[b_{sj} - \{1 + 2p_j\alpha^{-1} + b_{jj}(\underset{\sim}{p})\}\bar{d}_{sj}]$$

$$x[a_{sj} - a_{jj}(\underset{\sim}{p})\bar{d}_{sj}]$$

and

$$d_{ss}(\underset{\sim}{p} + \alpha\underset{\sim}{\delta}_j) = d_{ss}(\underset{\sim}{p}) - \bar{d}_{sj}d_{sj}$$

thus completing the updating operations.

## REFERENCES

BOX, G.E.P. and DRAPER, N.R. (1959). A basis for the selection of a response surface design. J. Amer. Statist. Assoc., 54, 622-654.

- (1963). The choice of a second order rotatable design. Biometrika, 50, 335-352.

FEDOROV, V.V. (1972). Theory of optimal experiments, ed. and trans. W.J. Studden and E.M. Klimko, New York: Academic Press.

GALIL, Z. and KIEFER, J. (1977). Comparison of Box-Draper and D-optimum designs for experiments with mixtures. Technometrics, 19, 441-444.

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

HUBER, P.J. (1975). Robustness and designs. In A survey of statistical design and linear models, ed. J.N. Srivastava, Amersterdam: North Holland.

KARSON, M.J., MANSON, A.R. and HADER, R.J. (1969). Minimum bias estimation and experimental design for response surfaces. Technometrics, 11, 461-475.

KUPPER, L.L. and MEYDRECH, E.F. (1974). Experimental design considerations based on a new approach to mean square error estimation of response surfaces. J. Amer. Statist. Assoc., 69, 461-463.

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

NAG, (1978). NAG Fortran Library Manual Mark 7, Vol. 2. Oxford: NAG.

SILVEY, S.D. (1980). Optimal design: an introduction to the theory for parameter estimation. London: Chapman and Hall.

STIGLER, S.M. (1971). Optimal experimental design for polynomial regression. J. Amer. Statist. Assoc., 66, 311-318.

WELCH, W.J. (1981). Computer-aided design of experiments for response surface estimation. Submitted for publication.

Table 1. Optimal approximate designs and properties.

| $nz^2_{max}\sigma^{-2}$ | Optimal design $p_1$ | $p_2$ | $p_3$ | $V(\underset{\sim}{p})$ | $B^{**}(\underset{\sim}{p})$ | $J^{**}(\underset{\sim}{p})$ | $e(\underset{\sim}{p},0)$ | $e(\underset{\sim}{p},\infty)$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.25 | 0 | 0 | 2.33 | 0.917 | 2.33 | 100 | 72.7 |
| 2 | 0.18 | 0.07 | 0 | 2.55 | 0.734 | 4.02 | 91.5 | 90.9 |
| 8 | 0.14 | 0.11 | 0.02 | 2.74 | 0.681 | 8.19 | 85.3 | 97.8 |
| $\infty$ | 0.11 | 0.11 | 0.11 | 3.00 | 0.667 | $\infty$ | 77.8 | 100 |

Table 2. Optimal exact designs and properties (n = 8)

| $z_{max}\sigma^{-1}$ | Optimal design $p_1$ | $p_2$ | $p_3$ | $V(\underset{\sim}{p})$ | $B^{**}(\underset{\sim}{p})$ | $e(\underset{\sim}{p},0)$ | $e(\underset{\sim}{p},\infty)$ |
|---|---|---|---|---|---|---|---|
| $<0.49$ | 0.25 | 0 | 0 | 2.33 | 0.917 | 100 | 74.2 |
| $\geq 0.49$ | 0.125 | 0.125 | 0 | 2.78 | 0.681 | 84.0 | 100 |

# PART 4

# FORTRAN LISTING OF ALGORITHMS

# ACED (ALGORITHMS FOR THE CONSTRUCTION OF EXPERIMENTAL DESIGNS)

## USER'S GUIDE

## 1. PURPOSE

ACED IMPLEMENTS THE ALGORITHMS DESCRIBED IN THE THREE PAPERS:

(P1) BRANCH AND BOUND SEARCH FOR EXPERIMENTAL DESIGNS BASED UPON D-OPTIMALITY AND OTHER CRITERIA

(P2) COMPUTER-AIDED DESIGN OF EXPERIMENTS FOR RESPONSE SURFACE ESTIMATION

(P3) A MEAN SQUARE ERROR CRITERION FOR THE DESIGN OF EXPERIMENTS.

THESE PAPERS CONTAIN THE ESSENTIAL BACKGROUND AND DESCRIPTION OF THE ALGORITHMS. ACED WORKS WITH FOUR DESIGN CRITERIA:

(C1) D-OPTIMALITY, ABBREVIATED AS "DO" IN ACED (SEE PAPER (P1)).

(C2) MINIMIZE THE AVERAGE VARIANCE OF THE RESPONSE SURFACE ESTIMATOR OVER THE DESIGN REGION. ALSO REFERRED TO AS V-OPTIMALITY IN PAPER (P2), BUT ABBREVIATED AS "AV" IN ACED.

(C3) MINIMIZE THE MAXIMUM VARIANCE OF THE RESPONSE SURFACE ESTIMATOR OVER THE DESIGN REGION. ALSO COMMONLY CALLED G-OPTIMALITY, BUT ABBREVIATED AS "MV" IN ACED. SEE PAPER (P2).

(C4) MINIMIZE THE AVERAGE MEAN SQUARE ERROR OF THE RESPONSE SURFACE ESTIMATOR, WITH THE AVERAGE TAKEN OVER THE DESIGN REGION AND POTENTIAL BAD DEPARTURES FROM THE ASSUMED MODEL. ALSO REFERRED TO AS J** IN PAPER (P3), BUT ABBREVIATED AS "AM" IN ACED.

THREE ALGORITHMS ARE AVAILABLE FOR SUBSETS OF THESE CRITERIA:

(A1) BRANCH AND BOUND, ABBREVIATED AS "BR" IN ACED. SEE PAPER (P1). ONLY AVAILABLE WITH DO CRITERION.

(A2) EXCURSION, ABBREVIATED AS "EX" IN ACED. THIS ALGORITHM IS A GENERALIZATION OF MITCHELL'S DETMAX (TECHNOMETRICS, 1974) TO ALLOW ALL FOUR OF THE ABOVE CRITERIA. SEE PAPERS (P2) AND (P3).

(A3) APPROXIMATE, ABBREVIATED AS "AP" IN ACED. THIS FINDS OPTIMAL APPROXIMATE (NOT NECESSARILY INTEGER REPLICATION) DESIGNS FOR THE DO, AV AND AM CRITERIA. SEE PAPERS (P1) AND (P3). IT CAN BE USED IN ITS OWN RIGHT, BUT IT ALSO PROVIDES THE SECOND BOUND FOR THE BRANCH AND BOUND ALGORITHM.

## 2. IMPLEMENTING THE PROGRAM

2.1 PLEASE NOTE THE EDITING COMMENTS AT THE BEGINNING OF THE LISTING.

2.2 IT IS POSSIBLE TO RUN ACED INTERACTIVELY OR BATCH, SO SET VARIABLE INTER IN FIRST DATA STATEMENT AS INDICATED IN DESCRIPTION OF VARIABLES. IN THE BATCH MODE COMMANDS ARE READ FROM A FILE; SOME COMMENTS ON THE PREPARATION OF THE FILE ARE GIVEN IN SECTION 3.

2.3 THE PROGRAM WAS WRITTEN IN CDC FORTRAN 4. A CDC EXTENSION THAT WILL NEED REPLACING BEFORE USE ON OTHER MACHINES IS THE MASS STORAGE FACILITY IN SUBROUTINE MS7MAN TO WRITE/READ TO/FROM A RANDOM ACCESS FILE.
HOLLERITH STRINGS ARE ENCLOSED IN QUOTES.
FREE FORMAT IS INDICATED BY * REPLACING THE FORMAT STATEMENT NUMBER.

2.4 SEVERAL NAG7F ROUTINES ARE EMPLOYED:
G05CCF IN SUBROUTINE INITDES TO RANDOMLY INITIALIZE THE RANDOM NUMBER GENERATOR
G05DAF IN SUBROUTINE INITDES IS A UNIFORM RANDOM NUMBER GENERATOR
F01ABF IN SUBROUTINE INFINV INVERTS A POSITIVE DEFINITE MATRIX
F03ABF IN SUBROUTINE INFMAT FINDS THE DETERMINANT OF A POSITIVE DEFINITE MATRIX
M01ABF IN SUBROUTINE ORDER RANKS THE ELEMENTS OF A VECTOR
E04ABF IN SUBROUTINE OPTRAN MINIMIZES A FUNCTION BY QUADRATIC INTERPOLATION.

2.5 IF YOU DO NOT REQUIRE ALL OF THE ALGORITHMS, ONLY CERTAIN SUBROUTINES NEED BE COMPILED. THE FOLLOWING ARE REQUIRED BY ALL ALGORITHMS:
MAIN PROGRAM, TITLE, REGION, NOBS, MODEL, XTOFFST XTOFSND, CRITER, ALGOR, INITDES, INITARR, INFMAT INFINV, DET, MATVEC, QUAD, UPDATE, PREPOST, OUTPROP, OUTDES, MS7MAN, PURPROP, WSMAT, MORE, FAIL.
TO IMPLEMENT THE ALGORITHMS THE FOLLOWING ARE ALSO NECESSARY.
FOR AP WE NEED:
APPROX, LOWBND, MING, OPTRAN, FUNCT1, SQBIAS.
FOR EX WE NEED:
EXCURS, ADDSUB, ORDER, CRITV, SQBIAS.
FOR BR WE NEED:
APPROX, LOWBND, MING, OPTRAN, FUNCT1, BANDB, WRNODE, BLIST, STACK, UPCON, SYMS, WRSYM, EXTCON.

2.6 INPUT/OUTPUT DEVICES. LOGICAL DEVICE 1 IS THE INPUT MEDIUM FOR ALL COMMANDS; DEVICE 2 IS USED FOR ALL WRITING TO THE USER; DEVICE 7 IS A MASS STORAGE FILE USED BY SUBROUTINE MS7MAN (SEE 2.3 ABOVE); AND DEVICE 8 IS A SEQUENTIAL FILE THAT SUBROUTINE STACK WILL WRITE ON AND READ FROM.

## 3. INPUT

IT IS SUGGESTED THAT ACED IS INITIALLY USED IN THE INTERACTIVE MODE, WHERE SELF-EXPLANATORY PROMPTS ARE GIVEN FOR THE INPUT REQUIREMENTS. WHEN THE NECESSARY COMMANDS ARE UNDERSTOOD, THE FILE FOR THE BATCH MODE MAY BE PREPARED SIMPLY BY LISTING THE SAME COMMANDS. SPACES (NOT COMMAS) SEPARATE INPUT ITEMS. START RESPONSES IN COLUMN 1.

THE SPECIFICATION OF THE ASSUMED LINEAR MODEL NEEDS SOME FURTHER EXPLANATION. THE AVAILABLE OPTIONS ARE A FIRST-ORDER MODEL (CONSTANT TERM AND MAIN EFFECTS), A SECOND-ORDER MODEL (CONSTANT TERM, MAIN EFFECTS, ALL SIMPLE INTERACTIONS BETWEEN TWO VARIABLES, AND SQUARED TERMS FOR ALL VARIABLES), OR A USER-DEFINED MODEL. THE LAST REQUIRES THE USER TO WRITE A SUBROUTINE CALLED XTOF AND MAKE IT AVAILABLE ON A LIBRARY. FOR EXAMPLE, A MODEL FOR TWO EXPLANATORY VARIABLES MIGHT INCLUDE A CONSTANT, THE TWO MAIN EFFECTS AND THE INTERACTION BETWEEN THE TWO VARIABLES. THEREFORE, IT HAS FOUR UNKNOWN PARAMETERS. THE SUBROUTINE WOULD BE WRITTEN AS FOLLOWS.

```
      SUBROUTINE XTOF (X, NV, F, MAXK, K)
C
C        ON ENTRY AND EXIT NV IS THE NUMBER OF VARIABLES
C        (2 IN OUR CASE)
C        ON ENTRY AND EXIT X OF DIMENSION NV CONTAINS
C        THE VALUES OF THE NV VARIABLES FOR AN ARBITRARY
C        POINT IN THE DESIGN REGION
C        ON ENTRY AND EXIT MAXK IS THE MAXIMUM NUMBER
C        OF PARAMETERS FOR WHICH ACED HAS BEEN
C        COMPILED
C        ON EXIT K WILL BE THE NUMBER OF PARAMETERS
C        (4 IN OUR CASE)
C        ON EXIT F OF DIMENSION MAXK WILL CONTAIN THE
C        VALUES OF THE K FUNCTIONS OF X
C
      DIMENSION X(NV), F(MAXK)
C
C        WE HAVE 4 PARAMETERS IN THIS EXAMPLE
C
      K = 4
C
C        FIRST TERM IN MODEL IS THE CONSTANT
C
      F(1) = 1.0
C
C        SECOND AND THIRD TERMS ARE THE MAIN EFFECTS
C
      F(2) = X(1)
      F(3) = X(2)
C
C        LAST TERM IS THE INTERACTION
C
      F(4) = X(1) * X(2)
C
      RETURN
      END
```

# FORTRAN LISTING

```
      PROGRAM ACED (INPUT = 128, OUTPUT = 128, TAPE7 = 128,
     &   TAPE8 , TAPE1 = INPUT, TAPE2 = OUTPUT)
C
C     ********************************************************
C     ACED (ALGORITHMS FOR CONSTRUCTION OF EXPERIMENTAL DESIGNS)
C     WRITTEN BY
C     WILLIAM J. WELCH, IMPERIAL COLLEGE, LONDON
C     FOR
C     WARREN SPRING LABORATOPY, STEVENAGE, ENGLAND.
C     VERSION 1.2 (IMPROVED UPDATE ROUTINES)
C     ********************************************************
C
C     PROGRAM SHOULD BE EDITED PRIOR TO COMPILATION TO DEFINE LARGEST
C     PROBLEM FOR WHICH ARRAY DIMENSIONS ARE SUFFICIENT.  REPLACE
C     FOLLOWING BY INTEGER CONSTANTS THROUGHOUT:
C
C     MAXNREG MAXIMUM NUMBER OF CANDIDATE POINTS IN THE DESIGN REGION
C     MAXNV   MAXIMUM NUMBER OF EXPLANATORY VARIABLES
C             (80 COLUMN OUTPUT ALLOWS NO MORE THAN 8,
C             120 COLUMN OUTPUT ALLOWS NO MORE THAN 14)
C     MAXK    MAXIMUM NUMBER OF PARAMETERS IN LINEAR MODEL
C     MKP1    SET TO MAXK + 1
C     MAXNP   MAXIMUM NUMBER OF PROTECTED OBSERVATIONS
C     MAXNDES MAXIMUM NUMBER OF DESIGNS THAT CAN BE DISPLAYED
C             AND STORED FOR FURTHER INSPECTION
C     MAXNDP1 SET TO MAXNDES + 1
C     MAXNFC  MAXIMUM NUMBER OF CRITERION VALUES OF FAILED
C             DESIGNS THAT CAN BE STORED BY EXCURSION ALGORITHM
C             (200 HAS BEEN FOUND SATISFACTORY)
C     MAXNSYM MAXIMUM NUMBER OF SYMMETRIES STORED BY BRANCH AND
C             BOUND ALGORITHM ( SET TO MAXNV * (MAXNV + 1) / 2 )
C
C     THE FOLLOWING SHOULD ALSO BE REPLACED BY REAL CONSTANTS
C     THROUGHOUT:
C
C     EPS - SMALL POSITIVE VALUE (1.0E-6 HAS BEEN SATISFACTORY)
C           THE REAL A IS ASSUMED LESS THAN OR EQUAL TO ZERO IF A < EPS
C     BIG - LARGE POSITIVE VALUE (1.0E50 HAS BEEN SATISFACTORY)
C
C     DESCRIPTION OF VARIABLES
C     ------------------------
C
C     NREG    NUMBER OF CANDIDATE POINTS IN DESIGN REGION
C     NV      NUMBER OF EXPLANATORY (X) VARIABLES
C     XREG    XREG(L, J) IS VALUE OF EXPLANATORY VARIABLE J FOR
C             CANDIDATE L (L = 1, NREG; J = 1, NV)
C     INTER   INTERACTIVE / BATCH INDICATOR
C             1 INTERACTIVE MODE, 0 BATCH MODE
C     N       NUMBER OF OBSERVATIONS
C     DESN    EXPERIMENTAL DESIGN; DESN(L) IS PROPORTION OF OBSERVATIONS
C             AT CANDIDATE L (L = 1, NREG).  NOTE NORMALIZATION OF DESIGN
C             DESN(1) + ... + DESN(NREG) = 1
C     CONS    CONS(1, L) AND CONS(2, L) ARE LOWER AND UPPER CONSTRAINTS
C             ON DESN(L) (L = 1, NREG)
C             0 <= CONS(1, L) <= DESN(L) <= CONS(2, L) <= 1
C     IMOD    INDICATOR FOR LINEAR MODEL
C             1HF   FIRST-ORDER MODEL
C             1HS   SECOND-ORDER MODEL
C             1HU   USER-DEFINED MODEL VIA USER-WRITTEN SUBROUTINE
C                   XTOF (SEE USER'S GUIDE)
C     K       NUMBER OF PARAMETERS IN LINEAR MODEL
C     FREG    FREG(L, J) IS COEFFICIENTOF PARAMETER J IN LINEAR MODEL
C             FOR CANDIDATE L (L = 1, NREG; J = 1, K)
C     ICRIT   INDICATOR FOR CRITERION
C             2HDO   D-OPTIMALITY
C             2HAV   AVERAGE VARIANCE OF RESPONSE ESTIMATORS (V-OPTIMALITY)
C             2HMV   MAXIMUM VARIANCE OF RESPONSE ESTIMATORS (G-OPTIMALITY)
C             2HAM   AVERAGE MEAN SQUARE ERROR OF RESPONSE ESTIMATORS
C                    (AVERAGED OVER DESIGN REGION AND POTENTIAL BAD
C                    DEPARTURES FROM MODEL)
```

```
C         RATIO    IF ICRIT = 2HAM DEFINED AS RATIO OF MAXIMUM ABSOLUTE
C                  DEVIATION OF TRUE RESPONSE FROM ASSUMED MODEL TO ERROR
C                  "TANDARD DEVIATION
C         SCALEB   IF ICRIT = 2HAM DEFINED AS SCALEB = RATIO * RATIO / N,
C                  MULTIPLIER FOR AVERAGE SQUARE BIAS COMPONENT (AVESQB)
C                  IN AVERAGE MEAN SQUARE ERROR
C         IALG     INDICATOR FOR ALGORITHM
C                  2HAP    APPROXIMATE ALGORITHM
C                  2HEX    EXCURSION ALGORITHM
C                  2HBR    BRANCH AND BOUND ALGORITHM
C         NDES     NUMBER OF DESIGNS PRODUCED
C                  IF IALG = 2HAP THEN NDES = 1 (ALWAYS CONVERGES TO OPTIMUM)
C                  IF IALG = 2HEX THEN NDES IS NUMBER OF ATTEMPTS FROM
C                  DIFFERENT RANDOM STARTING DESIGNS
C                  IF IALG = 2HBR THEN AT EXIT FROM BRANCH AND BOUND ALGORITHM
C                  SUBROUTINE BANDB, NDES IS NUMBER OF DESIGNS ACTUALLY FOUND
C         IOUT     IF IALG = 2HBR THEN DEFINED AS INDICATOR FOR OUTPUT
C                  1HY    OUTPUT FOR EVERY NODE
C                  1HN    NO NODE OUTPUT
C         GAMMA    IF IALG = 2HBR THEN ALL DESIGNS WITH DETPAR (DETERMINANT
C                  OF COVARIANCE MATRIX OF THE PARAMETER ESTIMATORS) LESS
C                  THAN (1 + GAMMA) * D-OPTIMAL DETPAR ARE SUPPLIED SUBJECT
C                  TO A STORAGE LIMITATION OF MAXNDES DESIGNS
C         COVPAR   COVPAR(I, J) IS THE COVARIANCE BETWEEN ESTIMATORS FOR
C                  PARAMETERS I AND J (I = 1, K; J = 1, K) BUT NORMALIZED
C                  FOR N AND (UNSPECIFIED) ERROR VARIANCE.  INVERSE OF
C                  INFORMATION MATRIX.
C         DETPAR   DETERMINANT OF COVPAR
C         VAR      VAR(L) IS VARIANCE OF RESPONSE ESTIMATOR AT CANDIDATE L
C                  (L = 1, NREG) NORMALIZED FOR N AND ERROR VARIANCE
C         U        INFORMATION MATRIX FOR UNIFORM DESIGN
C                  U(I, J) = ( FREG(1, I) * FREG(1, J) + ...
C                  + FREG(NREG, I) * FREG(NREG, J) ) / NREG
C                  (I = 1, K; J = 1, K)
C         CUC      K X K MATRIX COVPAR * U * COVPAR
C         FCUCF    FCUCF(L) = F TRANSPOSE * CUC * F WHERE F IS TRANSPOSE
C                  OF ROW L OF FREG (L = 1, NREG)
C         AVEVAR   AVERAGE VARIANCE OF RESPONSE SURFACE ESTIMATORS
C         ALPHA    ADJUSTMENT COEFFICIENT TO AVOID SINGULARITY AS DESCRIBED
C                  BY MITCHELL, 1974, TECHNOMETRICS
C         NUPD     NUMBER OF TIMES WORKING ARRAYS ARE UPDATED
C         NODBEST  NODBEST(M) IS NODE AT WHICH DESIGN M IS FOUND BY BRANCH
C                  AND BOUND ALGORITHM SUBROUTINE BANDB (M = 1, NDES)
C         IREPLY   INDICATOR FOR START AGAIN WITH SAME DESIGN REGION
C                  1HY    START AGAIN
C                  1HN    STOP
C         AVESQB   AVERAGE SQUARE BIAS OF RESPONSE ESTIMATORS (AVERAGED
C                  OVER DESIGN REGION AND POTENTIAL BAD DEPARTURES
C                  FROM ASSUMED MODEL)
C         QQ       IF ICRIT = 2HAM THEN K X K MATRIX QQ = COVPAR * FREG
C                  * DIAG(DESN) * DIAG(DESN) * TRANSPOSE OF FREG * COVPAR
C         FQQF     FQQF(L) = F TRANSPOSE * QQ * F WHERE F IS TRANSPOSE
C                  OF ROW L OF FREG (L = 1, NREG)
C         QQUC     IF ICRIT = 2HAM THEN K X K MATRIX QQUC = QQ * U * COVPAR
C         FQQUCF   FQQUCF(L) = F TRANSPOSE * QQUC * F  WHERE F IS TRANSPOSE
C                  OF ROW L OF FREG (L = 1, NREG)
C         IFAIL    FAILURE INDICATOR
C                  0          O.K.
C                  NONZERO    FAILURE OCCURED
C
      DIMENSION XREG(MAXNREG, MAXNV), CONS(2, MAXNREG),
     &   FREG(MAXNREG, MAXK), DESN(MAXNREG), COVPAR(MAXK, MAXK),
     &   VAR(MAXNREG), CUC(MAXK, MAXK), FCUCF(MAXNREG), NODBEST(MAXNDES)
C
C
      COMMON /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &   QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
      DATA INTER, ALPHA, IFAIL /0,. 0.01, 0/
C
      REWIND 1
      REWIND 2
      REWIND 7
C
C        WRITE TITLE
C
      CALL TITLE
```

```
C
C        READ NREG, NV AND XREG
C
      CALL REGION (NREG, NV, XREG, INTER)
C
C        READ N AND CONS
C
    3 CALL NOBS (N, CONS, NREG, INTER)
C
C        READ IMOD, K AND FREG
C
      CALL MODEL (XREG, NREG, NV, IMOD, K, FREG, INTER, N)
C
C        READ ICRIT AND, IF ICRIT = 2HAM, RATIO
C
      CALL CRITER (ICRIT, RATIO, INTER)
      IF (ICRIT .EQ. 2HAM) SCALEB = RATIO * RATIO * FLOAT(N)
C
C        READ IALG, NDES AND, IF IALG = 2HBR, IOUT AND GAMMA
C
      CALL ALGOR (IALG, ICRIT, NDES, IOUT, GAMMA, INTER)
      IF (IALG .EQ. 2HAP) GO TO 20
      IF (IALG .EQ. 2HEX) GO TO 30
      IF (IALG .EQ. 2HBR) GO TO 40
      CALL FAIL (1)
      STOP
C
C        APPROXIMATE ALGORITHM
C        ---------------------
C        INITIALIZE DESIGN
C
C
   20 CALL INITDES (CONS, NREG, K, N, DESN, 0, 0, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 22
      CALL FAIL (IFAIL)
      STOP
C
C        INITIALIZE WORKING ARRAYS AND SCALARS FOR CHOSEN CRITERION
C
   22 CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, 0.0, N, ICRIT, 0, 0, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 24
      CALL FAIL (IFAIL)
      STOP
   24 NUPD = 0
C
C        FIND OPTIMAL APPROXIMATE DESIGN
C
      CALL APPROX (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, ICRIT, DUMMY, NUPD,
     &   0.0, IALG, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 26
      CALL FAIL (IFAIL)
      STOP
C
C        OUTPUT DESIGN PROPERTIES
C
   26 CALL OUTPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, 1, 1, ICRIT, IMOD, IALG, 0.0, RATIO,
     &   NUPD / 2, N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 28
      CALL FAIL (IFAIL)
      STOP
C
C        OUTPUT DESIGN
C
   28 CALL OUTDES (80, 1, DESN, NREG, N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 50
      CALL FAIL (IFAIL)
      STOP
```

```
C
C        EXCURSION ALGORITHM
C        -------------------
C
   30 DO 32 II = 1, NDES
C
C        FIND NEAR-OPTIMAL EXACT DESIGN
C
      CALL EXCURS (CONS, NREG, FREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, ICRIT, NUPD, N, ALPHA, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 31
      CALL FAIL (IFAIL)
      STOP
C
C        OUTPUT PROPERTIES OF DESIGN II
C
   31 CALL OUTPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, II, NDES, ICRIT, IMOD, IALG, 0.0, RATIO,
     &   NUPD, N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 32
      CALL FAIL (IFAIL)
      STOP
C
C        STORE DESIGN II
C
   32 CALL MS7MAN (II, DESN, NREG, 1)
C
      DO 34 II = 1, NDES
C
C        GET DESIGN II
C
      CALL MS7MAN (II, DESN, NREG, -1)
C
C        OUTPUT DESIGN II
C
      CALL OUTDES (80, II, DESN, NREG, N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 34
      CALL FAIL (IFAIL)
      STOP
   34 CONTINUE
      GO TO 50
C
C         BRANCH AND BOUND ALGORITHM
C         --------------------------
C
C         FIND D-OPTIMAL AND NEARLY D-OPTIMAL EXACT DESIGNS
C
   40 CALL BANDB (CONS, XREG, NREG, NV, FREG, K, DESN, COVPAR,
     &   VAR, CUC, FCUCF, IOUT, NDES, N, ALPHA, GAMMA, NODBEST, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 42
      CALL FAIL (IFAIL)
      STOP
   42 DO 41 II = 1- NDES
C
C        GET DESIGN II
C
      CALL MS7MAN (II, DESN, NREG, -1)
C
C        OUTPUT PROPERTIES OF DESIGN II
C
      CALL OUTPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, II, NDES, ICRIT, IMOD, IALG, GAMMA, 0.0,
     &   NODBEST(II), N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 41
      CALL FAIL (IFAIL)
      STOP
   41 CONTINUE
C
      DO 44 II = 1, NDES
C
C        GET DESIGN II
C
      CALL MS7MAN (II, DESN, NREG, -1)
```

```
C
C        OUTPUT DESIGN II.
C
      CALL OUTDES (80, II, DESN, NREG, N, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 44
      CALL FAIL (IFAIL)
      STOP
   44 CONTINUE
C
C        OUTPUT FURTHER PROPERTIES OF SPECIFIED DESIGNS
C
   50 CALL FURPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   ICRIT, NDES, INTER, IFAIL)
      IF (IFAIL .EQ. 0) GO TO 60
      CALL FAIL (IFAIL)
      STOP
C
C
C        READ IREPLY AND START AGAIN WITH SAME DESIGN REGION IF IREPLY = 1HY
C
   60 CALL MORE (IREPLY, INTER)
      IF (IREPLY .EQ. 1HY) GO TO 3
      STOP
      END
C
C     --------------------
C
      SUBROUTINE TITLE
C
      WRITE (2, 100)
  100 FORMAT (// 6X, 58(1H*) /
     &   6X, "ACED (ALGORITHMS FOR CONSTRUCTION OF EXPERIMENTAL",
     &   " DESIGNS)" / 6X, "WRITTEN BY" /
     &   6X, "WILLIAM J. WELCH, IMPERIAL COLLEGE, LONDON" /
     &   6X, "FOR" /
     &   6X, "WARREN SPRING LABORATORY, STEVENAGE, ENGLAND" /
     &   6X, "VERSION 1.2" /
     &   6X, 58(1H*) )
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE REGION (NREG, NV, XREG, INTER)
C
C        READ NREG, NV AND XREG
C        (PARAMETER LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION XREG(MAXNREG, MAXNV), NLEV(MAXNV), AL(MAXNREG)
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
C        READ NREG
C
      DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ " ENTER NUMBER OF CANDIDATE POINTS (MAX MAXNREG)" /)
      READ (1, *) NREG
      IF (EOF(1) .EQ. 0.0) GO TO 3
      WRITE (2, 120)
  120 FORMAT (/ " ERROR: END OF FILE WHEN READING NUMBER OF",
     &   " CANDIDATES" )
      GO TO 4
    3 IF (NREG .GE. 2 .AND. NREG .LE. MAXNREG) GO TO 7
      WRITE (2, 122)
  122 FORMAT (/ " ERROR: NUMBER OF CANDIDATES LESS THAN 2 OR",
     &   " EXCEEDS MAXNREG" )
    4 CONTINUE
      STOP
C
C        READ NV
C
    7 DO 8 NN = 1,MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 101)
  101 FORMAT (/ " ENTER NUMBER OF EXPLANATORY VARIABLES (MAX MAXNV)" /)
      READ (1, *) NV
```

```
      IF (EOF(1) .EQ. 0.0) GO TO 9
      WRITE (2, 124)
  124 FORMAT (/ ' ERROR: END OF FILE WHEN READING NUMBER OF',
     &    ' EXPLANATORY VARIABLES' )
      GO TO 8
    9 IF (NV .GE. 1 .AND. NV .LE. MAXNV) GO TO 10
      WRITE (2, 126)
  126 FORMAT (/ ' ERROR: NUMBER OF EXPLANATORY VARIABLES LESS',
     &    ' THAN 1 OR EXCEEDS MAXNV' )
    8 CONTINUE
      STOP
C
C        READ FACTORIAL REPRESENTATION OR MISCELLANEOUS LIST
C
   10 DO 11 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 102)
  102 FORMAT (/ ' FACTORIAL REPRESENTATION OR MISCELLANEOUS',
     &    ' LIST OF DESIGN CANDIDATES (F OR M)' /)
      READ (1, 103) IREPLY
  103 FORMAT (A1)
      IF (EOF(1) .EQ. 0.0) GO TO 13
      WRITE (2, 128)
  128 FORMAT (/ ' ERROR: END OF FILE WHEN READING FACTORIAL',
     &    ' REPRESENTATION OR MISCELLANEOUS' / ' CANDIDATES' )
      GO TO 11
   13 IF (IREPLY .EQ. 1HF) GO TO 20
      IF (IREPLY .EQ. 1HM) GO TO 40
      WRITE (2, 130)
  130 FORMAT (/ ' ERROR: F OR M EXPECTED TO ANSWER FACTORIAL',
     &    ' REPRESENTATION OR MISCELLANEOUS' / ' CANDIDATES' )
   11 CONTINUE
      STOP
C
C        FACTORIAL ARRANGEMENT OF CANDIDATES
C
C        READ NUMBERS OF LEVELS OF EXPLANATORY VARIABLES
C
   20 DO 21 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 105) NV
  105 FORMAT (/ ' ENTER', I3, ' INTEGER(S) FOR THE NUMBER OF',
     &    ' LEVELS OF EACH VARIABLE' /)
      READ (1, *) (NLEV(I), I = 1, NV)
      IF (EOF(1) .EQ. 0.0) GO TO 24
      WRITE (2, 134)
  134 FORMAT (/ ' ERROR: END OF FILE WHEN READING NUMBERS OF LEVELS',
     &    ' OF EXPLANATORY VARIABLES' )
      GO TO 21
   24 NCHECK = 1
      DO 22 I = 1, NV
      IF (NLEV(I) .GE. 2 .AND. NLEV(I) .LE. NREG) GO TO 22
      WRITE (2, 136) I
  136 FORMAT (/ ' ERROR: EXPLANATORY VARIABLE', I3, ' HAS NUMBER',
     &    ' OF LEVELS LESS THAN 1 OR' / ' EXCEEDING NUMBER OF',
     &    ' CANDIDATES' )
      GO TO 21
   22 NCHECK = NCHECK * NLEV(I)
      IF (NCHECK .EQ. NREG) GO TO 30
      WRITE (2, 138)
  138 FORMAT (/ ' ERROR: NUMBERS OF LEVELS OF EXPLANATORY VARIABLES',
     &    ' INCONSISTENT WITH NUMBER' / ' OF CANDIDATES' )
   21 CONTINUE
      STOP
C
C        READ VALUES OF LEVELS AND GENERATE FACTORIAL CANDIDATES
C
   30 KK = 1
      MM = NREG
      DO 32 I = 1, NV
      DO 35 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 107) NLEV(I), I
  107 FORMAT (/ ' ENTER', I4, ' REALS FOR LEVELS (ACTUAL OR',
     &    ' TRANSFORMED) OF VARIABLE', I3 /)
      READ (1, *) (AL(J), J = 1, NLEV(I))
      IF (EOF(1) .EQ. 0.0) GO TO 36
      WRITE (2, 142) I
  142 FORMAT (/ ' ERROR: END OF FILE WHEN READING LEVELS OF',
     &    ' EXPLANATORY VARIABLE', I3 )
   35 CONTINUE
      STOP
```

```
   36 MM = MM / NLEV(I)
      IC = 0
      DO 33 KAA = 1, KK
      DO 33 J = 1, NLEV(I)
      DO 33 M = 1, MM
      IC = IC + 1
   33 XREG(IC, I) = AL(J)
   32 KK = KK * NLEV(I)
      GO TO 60
C
C        MISCELLANEOUS LIST
C
   40 DO 41 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 108) NREG, NV
  108 FORMAT (/ ' FOR EACH OF THE', I4, ' CANDIDATES ENTER',
     &    ' REAL(S) FOR THE VALUE OF EACH VARIABLE' /)
      READ (1, *) ((XREG(L, J), J = 1, NV), L = 1, NREG)
      IF (EOF(1) .EQ. 0.0) GO TO 60
      WRITE (2, 146)
  146 FORMAT (/ ' ERROR: END OF FILE WHEN READING MISCELLANEOUS',
     &    ' LIST OF CANDIDATES' )
   41 CONTINUE
      STOP
C
C        WRITE CANDIDATES IF REQUIRED
C
   60 DO 61 NN=1,MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 110)
  110 FORMAT (/ ' LIST OF CANDIDATES REQUIRED (YES OR NO)' /)
      READ (1, 103) IREPLY
      IF (EOF(1) .EQ. 0.0) GO TO 62
      WRITE (2, 150)
  150 FORMAT (/ ' ERROR: END OF FILE WHEN READING LIST OF CANDIDATES',
     &    ' REQUIRED' )
      GO TO 61
   62 IF (IREPLY .EQ. 1HN) RETURN
      IF (IREPLY .EQ. 1HY) GO TO 64
      WRITE (2, 152)
  152 FORMAT (/ ' ERROR: Y OR N EXPECTED TO ANSWER LIST OF',
     &    ' CANDIDATES REQUIRED' ).
   61 CONTINUE
      STOP
C
   64 WRITE (2, 112) NREG, NV, (I, I = 1, NV)
  112 FORMAT (/ ' CANDIDATE LIST - ', I4, ' POINTS' /
     &    1X, 28(1H-) // ' CANDIDATE', =(3X, 'VAR', I2))
      WRITE (2, 113)
  113 FORMAT (1H )
      DO 65 L = 1, NREG
   65 WRITE (2, 114) L, NV, (XREG(L, J), J = 1, NV)
  114 FORMAT (I6, 4X, =F8.3)
C
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE NOBS (N, CONS, NREG, INTER)
C
C        READ N AND CONS
C        (PARAMETER LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), IP(MAXNP)
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
C        READ N
C
      DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ ' NUMBER OF OBSERVATIONS' /)
      READ (1, *) N
      IF (EOF(1) .EQ. 0.0) GO TO 5
      WRITE (2, 120)
  120 FORMAT (/ ' ERROR: END OF FILE WHEN READING NUMBER OF',
     &    ' OBSERVATIONS' )
      GO TO 4
```

```
    5 IF (N .GE. 1) GO TO 7
      WRITE (2, 122)
  122 FORMAT (/ ' ERROR: NUMBER OF OBSERVATIONS LESS THAN 1' )
    4 CONTINUE
      STOP
C
C        READ NUMBER OF PROTECTED OBSERVATIONS NP
C
    7 DO 10 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 101)
  101 FORMAT (/ ' NUMBER OF PROTECTED OBSERVATIONS' /)
      READ (1, *) NP
      IF (EOF(1) .EQ. 0.0) GO TO 9
      WRITE (2, 126)
  126 FORMAT (/ ' ERROR: END OF FILE WHEN READING NUMBER OF',
     &    ' PROTECTED OBSERVATIONS' )
      GO TO 10
    9 IF (NP .GE. 0 .AND. NP .LE. N .AND. NP .LE. MAXNP) GO TO 11
      WRITE (2, 128)
  128 FORMAT (/ ' ERROR: NUMBER OF PROTECTED OBSERVATIONS LESS THAN 0,',
     &    ' EXCEEDS NUMBER OF' / ' OBSERVATIONS OR EXCEEDS MAXNP' )
   10 CONTINUE
      STOP
C
C        READ PROTECTED OBSERVATIONS
C
   11 IF (NP .EQ. 0) GO TO 19
      DO 12 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 102) NP
  102 FORMAT (/ ' ENTER', I4, ' INTEGER(S) REFERENCING THE',
     &    ' CANDIDATES FOR THE PROTECTED' / ' OBSERVATION(S)' /)
      READ (1, *) (IP(L), L=1, NP)
      IF (EOF(1) .EQ. 0.0) GO TO 15
      WRITE (2, 130)
  130 FORMAT (/ ' ERROR: END OF FILE WHEN READING PROTECTED',
     &    ' OBSERVATIONS' )
      GO TO 12
   15 DO 13 L = 1, NP
      IF (IP(L) .GE. 1 .AND. IP(L) .LE. NREG) GO TO 13
      WRITE (2, 132) L
  132 FORMAT (/ ' ERROR: PROTECTED OBSERVATION', I3, ' DOES NOT',
     &    ' REFERENCE CANDIDATE LIST' )
      GO TO 12
   13 CONTINUE
      GO TO 19
   12 CONTINUE
      STOP
C
C        INITIALIZE CONS(1, L) = 0, CONS(2, L) = 1 (L = 1, NREG)
C        AND INCREMENT CONS(1, L) BY 1 / N FOR EVERY PROTECTED OBSERVATION
C        AT CANDIDATE L.
C
   19 DO 25 L = 1, NREG
      CONS(1, L) = 0.0
   25 CONS(2, L) = 1.0
      IF (NP .EQ. 0) RETURN
      DO 20 L = 1, NP
      KAN = IP(L)
   20 CONS(1, KAN) = CONS(1, KAN) + 1.0 / FLOAT(N)
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE MODEL (XREG, NREG, NV, IMOD, K, FREG, INTER, N)
C
C        READ IMOD, K AND FREG
C        (PARAMETER LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION XREG(MAXNREG, NV), FREG(MAXNREG, MAXK),
     &    X(MAXNV), F(MAXK)
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
```

```
C
C        READ IMOD
C
      DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ ' MODEL: FIRST-ORDER, SECOND-ORDER OR',
     &    ' USER-DEFINED (USER HAS SUPPLIED' / ' SUBROUTINE XTOF).',
     &    '  ENTER F, S OR U' /)
      READ (1, 101) IMOD
  101 FORMAT (A1)
      IF (EOF(1) .EQ. 0.0) GO TO 15
      WRITE (2, 120)
  120 FORMAT (/ ' ERROR: END OF FILE WHEN READING MODEL' )
      GO TO 4
   15 IF (IMOD .EQ. 1HF) GO TO 20
      IF (IMOD .EQ. 1HS) GO TO 30
      IF (IMOD .EQ. 1HU) GO TO 40
      WRITE (2, 122)
  122 FORMAT (/ ' ERROR: F, S OR U EXPECTED TO ANSWER MODEL' )
      GO TO 4
C
C        CALCULATE OR GET K FOR MODEL
C
   20 K = NV + 1
      IF (INTER .EQ. 1) WRITE (2, 103) K
  103 FORMAT (/ ' FIRST-ORDER MODEL HAS', I3, ' PARAMETERS')
      GO TO 50
C
   30 K = (NV + 1) * (NV + 2) / 2
      IF (INTER .EQ. 1) WRITE (2, 104) K
  104 FORMAT (/ ' SECOND-ORDER MODEL HAS', I3, ' PARAMETERS')
      GO TO 50
C
C        USER DEFINED - CALL XTOF ONCE TO OBTAIN K
C
   40 DO 41 J = 1, NV
   41 X(J) = XREG(1, J)
      CALL XTOF (X, NV, F, MAXK, K)
C
C        CHECK K
C
   50 IF (K .GE. 1 .AND. K .LE. MAXK .AND. K. LE. N) GO TO 51
      WRITE (2, 126)
  126 FORMAT (/ ' ERROR: NUMBER OF PARAMETERS LESS THAN 1, EXCEEDS',
     &    ' NUMBER OF OBSERVATIONS' / ' OR EXCEEDS MAXK' )
    4 CONTINUE
      STOP
C
C        COMPUTE FREG
C
   51 DO 52 L = 1, NREG
      DO 53 J = 1, NV
   53 X(J) = XREG(L, J)
      IF (IMOD .EQ. 1HF) CALL XTOFFST (X, NV, F, K)
      IF (IMOD .EQ. 1HS) CALL XTOFSND (X, NV, F, K)
      IF (IMOD .EQ. 1HU) CALL XTOF (X, NV, F, MAXK, K)
      DO 52 J = 1, K
   52 FREG(L, J) = F(J)
C
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE XTOFFST (X, NV, F, K)
C
C        TRANSFORM X TO F FOR FIRST-ORDER MODEL
C
      DIMENSION X(NV), F(K)
C
      F(1) = 1.0
      DO 23 J = 1, NV
   23 F(J + 1) = X(J)
      RETURN
      END
```

```
      SUBROUTINE XTOFSND (X, NV, F, K)
C
C        TRANSFORM X TO F FOR SECOND-ORDER MODEL
C
      DIMENSION X(NV), F(K)
C
      F(1) = 1.0
      DO 23 J = 1, NV
   23 F(J + 1) = X(J)
      M = NV + 1
      DO 24 I = 1, NV
      DO 24 J = I, NV
      M = M + 1
   24 F(M) = X(I) * X(J)
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE CRITER (ICRIT, RATIO, INTER)
C
C        READ ICRIT AND, IF ICRIT = 2HAM, RATIO
C        (PARAMETER LIST DESCRIBED IN MAIN PROGRAM)
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
C        READ ICRIT
C
      DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ ' CRITERION: ENTER DO, AV, MV, AM OR HELP' /)
      READ (1, 101) ICRIT
  101 FORMAT (A2)
      IF (EOF(1) .EQ. 0.0) GO TO 2
      WRITE (2, 120)
  120 FORMAT (/ ' ERROR: END OF FILE WHEN READING CRITERION' )
      GO TO 3
    2 IF (ICRIT .EQ. 2HDO .OR. ICRIT .EQ. 2HAV
     &    .OR. ICRIT .EQ. 2HMV) RETURN
      IF (ICRIT .EQ. 2HAM) GO TO 20
      IF (ICRIT .NE. 2HHE) WRITE (2, 122)
  122 FORMAT (/ ' ERROR: DO, AV, MV OR AM EXPECTED TO ANSWER',
     &    ' CRITERION' )
    3 WRITE (2, 102)
  102 FORMAT (/ 1X, 'CRITERION', 46X, 'COMPATIBLE ALGORITHMS' //
     &    1X, 'DO', 9X, 'D-OPTIMALITY', 32X, 'ALL ALGORITHMS' //
     &    1X, 'AV', 9X, 'AVERAGE VARIANCE OF RESPONSE ESTIMATORS',
     &    5X, 'APPROXIMATE, EXCURSION' /
     &    12X, '(V-OPTIMALITY)', 30X, '(NOT BRANCH AND BOUND)' //
     &    1X, 'MV', 9X, 'MAXIMUM VARIANCE OF RESPONSE ESTIMATORS',
     &    5X, 'EXCURSION (NOT APPROXIMATE' /
     &    12X, '(G-OPTIMALITY)', 30X, 'OR BRANCH AND BOUND)' //
     &    1X, 'AM', 9X, 'AVERAGE MEAN SQUARE ERROR OF RESPONSE',
     &    7X, 'APPROXIMATE, EXCURSION' /
     &    12X, 'ESTIMATORS (AVERAGE OVER DESIGN REGION',
     &    6X, '(NO  BRANCH AND BOUND)' /
     &    12X, 'AND POTENTIAL BAD DEPARTURES FROM MODEL)' )
    4 CONTINUE
      STOP
C
C        READ RATIO (ICRIT = 2HAM)
C
   20 DO 5 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 103)
  103 FORMAT(/ ' RATIO OF MAX ABSOLUTE DEVIATION OF TRUE',
     &    ' RESPONSE FROM ASSUMED MODEL' / ' TO ERROR STANDARD',
     &    ' DEVIATION' /)
      READ (1, *) RATIO
      IF (EOF(1) .EQ. 0.0) GO TO 6
      WRITE (2, 124)
  124 FORMAT (/ ' ERROR: END OF FILE WHEN READING RATIO OF MAX',
     &    ' ABSOLUTE DEVIATION OF TRUE' / ' RESPONSE FROM ASSUMED',
     &    ' MODEL TO ERROR STANDARD DEVIATION' )
      GO TO 5
```

```
    6 IF (RATIO .GE. 0.0) RETURN
      WRITE (2, 126)
  126 FORMAT (/ ' ERROR: NEGATIVE RATIO OF MAX ABSOLUTE DEVIATION',
     &    ' OF TRUE RESPONSE FROM ASSUMED' / ' MODEL TO ERROR',
     &    ' STANDARD DEVIATION' )
    5 CONTINUE
      STOP
      END
C
C     ---------------------
C
      SUBROUTINE ALGOR (IALG, ICRIT, NDES, IOUT, GAMMA, INTER)
C
C        READ IALG AND
C        IF IALG = 2HAP THEN SET NDES = 1
C        IF IALG = 2HEX THEN READ NDES
C        IF IALG = 2HBR THEN READ GAMMA AND IOUT (NDES UNDEFINED)
C        (PARAMETER LIST DESCRIBED IN MAIN PROGRAM)
C
C        NOTE ONLY COMBINATIONS OF ALGORITHM AND CRITERION MARKED *
C        ARE ALLOWED
C
C                         ICRIT = 2HDO  ICRIT = 2HAV  ICRIT = 2HMV  ICRIT = 2HAM
C        IALG = 2HAP          *             *                         *
C        IALG = 2HEX          *             *             *           *
C        IALG = 2HBR          *
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
      IF (ICRIT .NE. 2HMV) GO TO 1
C
C        ICRIT = 2HMV SO ONLY IALG = 2HEX ALLOWED
C
      IF (INTER .EQ. 1) WRITE (2, 140)
  140 FORMAT (/' ONLY EXCURSION ALGORITHM ALLOWED FOR CRITERION MV' )
      IALG = 2HEX
      GO TO 10
C
C         READ IALG
C
    1 DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1 .AND. ICRIT .EQ. 2HDO) WRITE (2, 110)
  110 FORMAT (/ ' ALGORITHM: ENTER AP, EX, BR OR HELP' /)
      IF (INTER .EQ. 1 .AND. (ICRIT .EQ. 2HAV .OR. ICRIT .EQ. 2HAM))
     &   WRITE(2, 111)
  111 FORMAT (/ ' ALGORITHM: ENTER AP, EX OR HELP' /)
      READ (1, 101) IALG
  101 FORMAT (A2)
      IF (EOF(1) .EQ. 0.0) GO TO 2
      WRITE (2, 120)
  120 FORMAT (/ ' ERROR: END OF FILE WHEN READING ALGORITHM' )
      GO TO 3
    2 IF (IALG .EQ. 2HEX) GO TO 10
      IF (IALG .EQ. 2HAP) GO TO 15
      IF (IALG .EQ. 2HBR .AND. ICRIT .EQ. 2HDO) GO TO 20
      IF (IALG .NE. 2HHE) WRITE (2, 122)
  122 FORMAT (/ ' ERROR: ALGORITHM IS EITHER NOT AP, EX OR BR',
     &    ' OR IT IS INCOMPATIBLE' / ' WITH CRITERION' )
    3 WRITE (2, 102)
  102 FORMAT (/ ' ALGORITHM', 50X, 'COMPATIBLE CRITERIA' //
     &    ' APPROXIMATE', 8X, 'REPLICATIONS NOT NECESSARILY',
     &    ' INTEGER', 4X, 'DO, AV, AM (NOT MV)' //
     &    ' EXCURSION', 10X, 'GIVES NEAR-OPTIMAL DESIGN WITH',
     &    10X, 'ALL CRITERIA' /
     &    20X, 'INTEGER REPLICATIONS (GENERALIZATION' /
     &    20X, 'OF MITCHELL'S DETMAX)' //
     &    ' BRANCH AND BOUND', 3X, 'GIVES OPTIMAL DESIGN WITH',
     &    ' INTEGER', 7X, 'DO (NOT AV, MV, AM)' /
     &    20X, 'REPLICATION BUT EXPENSIVE COMPARED TO' /
     &    20X, 'EXCURSION' )
    4 CONTINUE
      STOP
```

```
C
C        APPROXIMATE ALGORITHM
C
   15 NDES = 1
      RETURN
C
C        EXCURSION ALGORITHM - READ NDES
C
   10 DO 11 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 103)
  103 FORMAT (/ ' NUMBER OF ATTEMPTS FROM DIFFERENT',
     &    ' STARTING DESIGNS (MAX MAXNDES)' /)
      READ (1, *) NDES
      IF (EOF(1) .EQ. 0.0) GO TO 12
      WRITE (2, 126)
  126 FORMAT (/ ' ERROR: END OF FILE WHEN READING NUMBER OF',
     &    ' ATTEMPTS FROM DIFFERENT STARTING' / ' DESIGNS' )
      GO TO 11
   12 IF (NDES .GE. 1 .AND. NDES .LE. MAXNDES) RETURN
      WRITE (2, 128)
  128 FORMAT (/ ' ERROR: NUMBER OF ATTEMPTS FROM DIFFERENT STARTING',
     &    ' DESIGNS LESS THAN 1' / ' OR EXCEEDS MAXNDES' )
   11 CONTINUE
      STOP
C
C        BRANCH AND BOUND ALGORITHM
C
C        READ GAMMA
C
   20 DO 21 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 104)
  104 FORMAT (/ ' BRANCH AND BOUND ALGORITHM SUPPLIES THE D-OPTIMAL',
     &    ' AND NEARLY D-OPTIMAL' /
     &    ' DESIGNS: ALL THOSE WITH DETPAR (DETERMINANT OF THE',
     &    ' COVARIANCE MATRIX OF THE' /
     &    ' PARAMETER ESTIMATORS) LESS THAN (1 + GAMMA) *',
     &    ' D-OPTIMAL DETPAR' /
     &    ' BUT SUBJECT TO STORAGE LIMITATION OF MAXNDES DESIGNS' /
     &    ' ENTER GAMMA' /)
      READ(1, *) GAMMA
      IF (EOF(1) .EQ. 0.0) GO TO 22
      WRITE (2, 132)
  132 FORMAT (/ ' ERROR: END OF FILE WHEN READING BRANCH AND BOUND',
     &    ' GAMMA' )
      GO TO 21
   22 IF (GAMMA .GE. 0.0) GO TO 60
      WRITE (2, 134)
  134 FORMAT (/ ' ERROR: NEGATIVE BRANCH AND BOUND GAMMA' )
   21 CONTINUE
      STOP
C
C        READ IOUT
C
   60 DO 65 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 109)
  109 FORMAT (/ ' OUTPUT FOR EVERY NODE (YES OR NO)' /)
      READ (1, 112) IOUT
  112 FORMAT (A1)
      IF (EOF(1) .EQ. 0.0) GO TO 64
      WRITE (2, 136)
  136 FORMAT (/ ' ERROR: END OF FILE WHEN READING BRANCH AND',
     &    ' BOUND OUTPUT OPTION' )
      GO TO 65
   64 IF (IOUT .EQ. 1HY .OR. IOUT .EQ. 1HN) RETURN
      WRITE (2, 138)
  138 FORMAT (/ ' ERROR: Y OR N EXPECTED TO ANSWER BRANCH AND BOUND',
     &    ' OUTPUT OPTION' )
   65 CONTINUE
      STOP
      END
```

```
      SUBROUTINE INITDES (CONS, NREG, K, N, DESN, NALLOC, IRAN, IFAIL)
C
C        INITIALIZE DESN
C
C        IRAN     0   INITIALIZE DESN TO APPROXIMATE DETERMINISTIC
C                     DESIGN SATISFYING CONSTRAINTS
C                     (APPROXIMATE AND BRANCH AND BOUND ALGORITHMS)
C                 1   INITIALIZE TO EXACT RANDOM DESIGN SATISFYING
C                     CONSTRAINTS (EXCURSION ALGORITHM)
C        NALLOC   IF IRAN = 1 THEN NALLOC IS THE NUMBER OF THE N DESIGN
C                 POINTS FIXED BY LOWER CONSTRAINTS OR RANDOMLY CHOSEN
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 191   DESIGN COMPLETELY SPECIFIED BY CONSTRAINTS
C                 192   NO DESIGN CAN SATISFY CONSTRAINTS
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), DESN(NREG)
C
      DATA INITIAL /0/
C
      ALLOC = 0.0
      DO 4 L = 1, NREG
      DESN(L) = CONS(1, L)
    4 ALLOC = ALLOC + CONS(1, L)
      IF (ALLOC .LT. 1.0 - EPS) GO TO 3
      IFAIL = 191
      RETURN
C
    3 IF (IRAN .EQ. 1) GO TO 200
C
C        APPROXIMATE DETERMINISTIC DESIGN
C
      SLACK = 0.0
      DO 12 L = 1, NREG
   12 SLACK = SLACK + CONS(2, L) - CONS(1, L)
      IF (ALLOC + SLACK .GT. 1.0 - EPS) GO TO 15
      IFAIL = 192
      RETURN
   15 DO 16 L = 1, NREG
   16 DESN(L) = DESN(L) + (1.0 - ALLOC) * (CONS(2, L) - CONS(1, L))
     &   / SLACK
      RETURN
C
C        EXACT RANDOM DESIGN
C
  200 RN = 1.0 / FLOAT(N)
      NALLOC = IFIX ( FLOAT(N) * ALLOC )
      IF (INITIAL .EQ. 0) CALL G05CCF
      INITIAL = 1
      IUPRAN = K / 2
      IF (K .EQ. 1) IUPRAN = 1
      IF (NALLOC + IUPRAN .GT. N) IUPRAN = N - NALLOC
      NRAN = IFIX (G05DAF (1.0, FLOAT(IUPRAN) + 0.99))
      NALLOC = NALLOC + NRAN
      DO 6 MM = 1, NRAN
      II = IFIX (G05DAF (1.0, FLOAT(NREG) + 0.99))
      DO 8 L = 1, NREG
      LL = II + L
      IF (LL .GT. NREG) LL = LL - NREG
      IF (CONS(2, LL) - DESN(LL) .GT. RN - EPS) GO TO 9
    8 CONTINUE
      IFAIL = 192
      RETURN
    9 DESN(LL) = DESN(LL) + RN
    6 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, ALPHA, N, ICRIT, IALL, IEX, IFAIL)
C
C        INITIALIZE WORKING ARRAYS AND SCALARS FOR INITIAL DESIGN DESN
C        AS NECESSARY FOR CRITERION / ALGORITHM
C
C        IALL     1   COMPUTE DETPAR, VAR, CUC, AVEVAR IRRESPECTIVE OF
C                     CRITERION
C                 0   ONLY COMPUTE THESE IF NEEDED BY CRITERION
C        IEX      1   COMPUTE VAR IRRESPECTIVE OF CRITERION (ALWAYS
C                     NEEDED FOR EXCURSION ALGORITHM)
C                 0   ONLY COMPUTE VAR IF NEEDED BY CRITERION
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 181   CRITERION IS NOT DO, AV, MV OR AM
C                 (ANY OTHER NONZERO VALUE MEANS FAILURE IN A SUBROUTINE
C                 CALLED FROM INITARR)
C        (OTHER PARAMETERS IN LIST AND COMMON /AM/ DESCRIBED IN MAIN PROGRAM)
C
C        CRITERION   NECESSARY TO COMPUTE
C
C        DO          COVPAR, DETPAR, VAR
C        AV          COVPAR, AVEVAR, CUC, FCUCF
C        MV          COVPAR, VAR (IEX = 1 ALWAYS FOR MV CRITERION)
C        AM          COVPAR, AVEVAR, CUC, FCUCF, AVESQB, QQ, FQQF, QQUC, FQQUCF
C
      DIMENSION FREG(MAXNREG, K), DESN(NREG), COVPAR(MAXK, K),
     &   VAR(NREG), CUC(MAXK, K), FCUCF(NREG), U(MAXK, MAXK),
     &   F(MAXK), UC(MAXK, MAXK)
C
      COMMON /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &              QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
C        COMMON /SPACE1/ USED ONLY TO REDUCE STORAGE, NOT FOR COMMUNICATION
C
      COMMON /SPACE1/ XTX(MKP1, MAXK)
C
      DATA KALU /0/
C
      IF (ICRIT .EQ. 2HDO .OR. ICRIT .EQ. 2HAV
     &   .OR. ICRIT .EQ. 2HMV .OR. ICRIT .EQ. 2HAM) GO TO 3
      IFAIL = 181
      RETURN
C
C        IF FIRST CALL COMPUTE U - INFORMATION MATRIX FOR UNIFORM DESIGN
C
    3 IF (KALU .EQ. 1) GO TO 5
      KALU = 1
      DO 7 I = 1, K
      DO 7 J = 1, I
      U(I, J) = 0.0
      DO 8 L = 1, NREG
    8 U(I, J) = U(I, J) + FREG(L, I) * FREG(L, J)
      U(I, J) = U(I, J) / FLOAT (NREG)
    7 U(J, I) = U(I, J)
C
C        COMPUTE COVPAR
C
    5 CALL INFMAT (FREG, NREG, K, DESN, XTX)
      ADJ = ALPHA / FLOAT(N)
      DO 9 I = 1, K
      DO 9 J = I, K
    9 XTX(I, J) = XTX(I, J) + ADJ * U(I, J)
      CALL INFINV (XTX, COVPAR, K, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      DO 10 I = 1, K
      DO 10 J = 1, I
   10 COVPAR(J, I) = COVPAR(I, J)
C
      IF (ICRIT .NE. 2HDO .AND. IALL .NE. 1) GO TO 14
C
C        COMPUTE DETPAR
C
      CALL DET (XTX, K, DETXTX, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      DETPAR = 1.0 / DETXTX
```

```
C
   14 IF (ICRIT .NE. 2HDO .AND. IALL .NE. 1
     &    .AND. IEX .NE. 1) GO TO 18
C
C        COMPUTE VAR
C
      DO 15 L = 1, NREG
      DO 16 J = 1, K
   16 F(J) = FREG(L, J)
   15 CALL QUAD (F, COVPAR, MAXK, K, VAR(L))
C
   18 IF (ICRIT .NE. 2HAV .AND. ICRIT .NE. 2HAM
     &    .AND. IALL .NE. 1) RETURN
C
C        COMPUTE UC = U * COVPAR
C
      DO 19 I = 1, K
      DO 19 J = 1, K
      UC(I, J) = 0.0
      DO 19 M = 1, K
   19 UC(I, J) = UC(I, J) + U(I, M) * COVPAR(M, J)
C
C        COMPUTE AVEVAR = TRACE UC
C
      AVEVAR = 0.0
      DO 21 I = 1, K
   21 AVEVAR = AVEVAR + UC(I, I)
C
C        COMPUTE CUC
C
      DO 26 I = 1, K
      DO 26 J = 1, I
      CUC(I, J) = 0.0
      DO 25 M = 1, K
   25 CUC(I, J) = CUC(I, J) + COVPAR(I, M) * UC(M, J)
   26 CUC(J, I) = CUC(I, J)
C
C        COMPUTE FCUCF
C
      DO 55 L = 1, NREG
      DO 56 J = 1, K
   56 F(J) = FREG(L, J)
   55 CALL QUAD (F, CUC, MAXK, K, FCUCF(L))
C
      IF (ICRIT .NE. 2HAM) RETURN
C
C        COMPUTE QQ
C
      DO 27 I = 1, K
      DO 27 J = 1, I
   27 QQ(I, J) = 0.0
      DO 28 L = 1, NREG
      WT = DESN(L) + ADJ / FLOAT(NREG)
      IF (WT .LT. EPS) GO TO 28
      DO 29 J = 1, K
   29 F(J) = FREG(L, J) * WT
      DO 30 I = 1, K
      DO 30 J = 1, I
   30 QQ(I, J) = QQ(I, J) + F(I) * F(J)
   28 CONTINUE
      DO 32 I = 1, K
      DO 32 J = 1, I
   32 QQ(J, I) = QQ(I, J)
      CALL PREPOST (QQ, COVPAR, K)
C
C        COMPUTE AVESQB
C
      AVESQB = 1.0 - 2.0 * FLOAT(K) / FLOAT(NREG)
      DO 38 I = 1, K
      DO 38 J = 1, K
   38 AVESQB = AVESQB + QQ(I, J) * U(I, J)
C
C        COMPUTE QQUC = QQ * UC
C
      DO 40 I = 1, K
      DO 40 J = 1, K
```

```
      QQUC(I, J) = 0.0
      DO 40 M = 1, K
   40 QQUC(I, J) = QQUC(I, J) + QQ(I, M) * UC(M, J)
C
C        REPLACE QQUC BY SYMMETRIC MATRIX (QQUC + TRANSPOSE OF QQUC) / 2
C
      DO 43 I = 1, K
      DO 43 J = 1, I
      QQUC(I, J) = (QQUC(I, J) + QQUC(J, I)) / 2.0
   43 QQUC(J, I) = QQUC(I, J)
C
C        COMPUTE FQQF AND FQQUCF
C
      DO 60 L = 1, NREG
      DO 61 J = 1, K
   61 F(J) = FREG(L, J)
      CALL QUAD (F, QQ, MAXK, K, FQQF(L))
   60 CALL QUAD (F, QQUC, MAXK, K, FQQUCF(L))
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE INFMAT (FREG, NREG, K, DESN, XTX)
C
C         COMPUTE INFORMATION MATRIX XTX FOR DESN
C
      DIMENSION FREG(MAXNREG, K), DESN(NREG), XTX(MKP1, K)
C
      DO 1 I = 1, K
      DO 1 J = I, K
    1 XTX(I, J) = 0.0
      DO 3 L = 1, NREG
      IF (DESN(L) .LT. EPS) GO TO 3
      DO 2 I = 1, K
      DO 2 J = I, K
    2 XTX(I, J) = XTX(I, J) + DESN(L) * FREG(L, I) * FREG(L, J)
    3 CONTINUE
C
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE INFINV (XTX, COVPAR, K, IFAIL)
C
C        INVERT XTX TO GIVE COVPAR
C
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                141   XTX NOT POSITIVE DEFINITE
C                142   XTX ILL-CONDITIONED
C
      DIMENSION XTX(MKP1, K), COVPAR(MAXK, K), WORKSP(MAXK)
C
      NAGFAIL = 1
      CALL F01ABF (XTX, MKP1, K, COVPAR, MAXK, WORKSP, NAGFAIL)
      IF (NAGFAIL .EQ. 1) IFAIL = 141
      IF (NAGFAIL .EQ. 2) IFAIL = 142
C
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE DET (XTX, K, DETXTX, IFAIL)
C
C        COMPUTE DETXTX - DETERMINANT OF XTX
C
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                151   XTX NOT POSITIVE DEFINITE
C                152   OVERFLOW - DETXTX TOO LARGE
C                153   UNDERFLOW - DETXTX TOO SMALL
C
      DIMENSION XTX(MKP1, K), WORKSP(MAXK)
C
```

```
      NAGFAIL = 1
      CALL F03ABF (XTX, MKP1, K, DETXTX, WORKSP, NAGFAIL)
      IF (NAGFAIL .EQ. 1) IFAIL = 151
      IF (NAGFAIL .EQ. 2) IFAIL = 152
      IF (NAGFAIL .EQ. 3) IFAIL = 153
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE MATVEC (S, V, IDIM, K, Z)
C
C        MULTIPLY K X K MATRIX S BY K X 1 VECTOR V TO GIVE
C        K X 1 VECTOR Z
C
      DIMENSION S(IDIM, K), V(K), Z(K)
C
      DO 30 I = 1, K
      Z(I) = 0.0
      DO 30 J = 1, K
   30 Z(I) = Z(I) + S(I, J) * V(J)
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE APPROX (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, ICRIT, CRITLOW, NUPD,
     &   BEST, IALG, IFAIL)
C
C        APPROXIMATE ALGORITHM
C
C        CRITLOW ON EXIT CRITLOW IS A LOWER BOUND ON THE OPTIMAL
C                CRITERION VALUE
C        BEST    IF IALG = 2HBR USED TO TEST FOR TERMINATION
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                101   ALGORITHM DOES NOT CONVERGE IN MAXIT ITERATIONS
C                102   CRITERION IS NOT DO, AV OR AM
C                (ANY OTHER NONZERO VALUE MEANS FAILURE IN A SUBROUTINE
C                CALLED FROM APPROX)
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
C        ALGORITHM IS SEQUENTIAL: EACH STEP INCREASES ONE ELEMENT OF
C        DESN AND DECREASES ANOTHER BY SAME AMOUNT TO REDUCE CRITERION.
C        TERMINATION OCCURS FOR
C        IALG = 2HAP WHEN CRITERION < (1 + TOL) * CRITLOW (I.E. CRITERION
C                    CLOSE TO CRITLOW
C        IALG = 2HBR WHEN EITHER CRITERION <= BEST (I.E. CURRENT DESIGN
C                    AS GOOD AS BEST) OR CRITLOW > BEST (NO DESIGN CAN
C                    BEAT BEST)
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K), DESN(NREG),
     &   COVPAR(MAXK, K), VAR(NREG), CUC(MAXK, K), FCUCF(NREG),
     &   G(MAXNREG)
C
      COMMON /AM/   AVESQB, SCALEB
C
      DATA TOL, MAXIT /0.001, 250/
C
      IF (ICRIT .EQ. 2HDO .OR. ICRIT .EQ. 2HAV
     &   .OR. ICRIT .EQ. 2HAM) GO TO 10
      IFAIL = 102
      RETURN
C
   10 CRITLOW = 0.0
      DO 1 NTRANS = 1, MAXIT
C
      IF (ICRIT .EQ. 2HDO) CRITVAL = DETPAR
      IF (ICRIT .EQ. 2HAV) CRITVAL = AVEVAR
      IF (ICRIT .EQ. 2HAM) CRITVAL = AVEVAR + SCALEB * AVESQB
C
C        RETURN IF BRANCH AND BOUND ALGORITHM AND
C        CRITVAL <= BEST (DESN AS GOOD AS BEST)
C
      IF (IALG .EQ. 2HBR .AND. CRITVAL .LT. (1.0 + EPS) * BEST)
     &   RETURN
```

```
C
C        COMPUTE CRITLOW AND GRADIENTS G
C
      CALL LOWBND (CONS, NREG, K, DESN, DETPAR,
     &   VAR, FCUCF, AVEVAR, ICRIT, CRITLOW, G, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        RETURN IF BRANCH AND BOUND ALGORITHM AND
C        CRITLOW > BEST (NO DESIGN CAN BEAT BEST)
C
      IF (IALG .EQ. 2HBR .AND. CRITLOW .GT. BEST) RETURN
C
C        RETURN IF APPROXIMATE ALGORITHM AND
C        CRITVAL < (1 + TOL) * CRITLOW (CRITVAL CONVERGED)
C
      IF (IALG .EQ. 2HAP .AND. CRITVAL .LT. (1.0 + TOL) * CRITLOW)
     &   RETURN
C
C        FIND CANDIDATES INC AND IDEC: DESN(INC) WILL BE INCREASED
C        AND DESN(IDEC) DECREASED
C
      CALL MING (CONS, NREG, DESN, INC, IDEC, G, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        FIND BETAOPT: AMOUNT OF TRANSFER FROM DESN(IDEC) TO DESN(INC)
C        SUCH THAT CRITERION IS MINIMIZED
      CALL OPTRAN (CONS, NREG, FREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, AVEVAR, INC, IDEC, BETAOPT, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        UPDATE DESN AND WORKING ARRAYS AND SCALARS
C
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, INC, BETAOPT, ICRIT, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, IDEC, -BETAOPT, ICRIT, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
    1 CONTINUE
C
      IF (IALG .EQ. 2HAP) IFAIL = 101
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE EXCURS (CONS, NREG, FREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, ICRIT, NUPD, N, ALPHA, IFAIL)
C
C        EXCURSION ALGORITHM: ON EXIT DESN IS A NEAR-OPTIMAL EXACT
C        DESIGN (BUT NOTE DESIGN PROPERTIES COVPAR, VAR AND CUC
C        ARE NOT DEFINED ON EXIT; THEY ARE PASSED ONLY AS WORKSPACE)
C
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                131   DESIGN COMPLETELY SPECIFIED BY CONSTRAINTS
C                132   CRITERION IS NOT DO, AV, MV OR AM
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGAM)
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K), DESN(NREG),
     &   COVPAR(MAXK, K), VAR(NREG), CUC(MAXK, K), FCUCF(NREG),
     &   FC(MAXNFC), BDESN(MAXNREG)
C
      COMMON /AM/   AVESQB, SCALEB
C
      DATA MAXNSUC, MAXFAIL, MAXLENG, MXSTRAY /50, 10, 200, 6/
C
      IF (ICRIT .EQ. 2HDO .OR. ICRIT .EQ. 2HAV
     &   .OR. ICRIT .EQ. 2HMV .OR. ICRIT .EQ. 2HAM) GO TO 2
      IFAIL = 132
      RETURN
C
    2 NUPD = 0
```

```
C
C        RANDOMIZE INITIAL DESIGN OF NCUR ( <= N ) POINTS
C
      CALL INITDES (CONS, NREG, K, N, DESN, NCUR, 1, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, ALPHA, N, ICRIT, 0, 1, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        INCREASE NCUR TO N
C
      IF (NCUR .EQ. N) GO TO 3
      INC = N - NCUR
      DO 4 JJJ = 1, INC
      CALL ADDSUB (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, 1, NCUR, N, ALPHA, NUPD, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
    4 CONTINUE
C
C        COMPUTE NBELOW AND NABOVE, THE EXCURSION DEVIATIONS
C        ALLOWED BY THE CONSTRAINTS AND MXSTRAY ABOVE AND BELOW N
C
    3 SIGC1 = 0.0
      SIGC2 = 0.0
      DO 8 L = 1, NREG
      SIGC1 = SIGC1 + CONS(1, L)
    8 SIGC2 = SIGC2 + CONS(2, L)
      NBELOW = IFIX ( (1.0 - SIGC1 + EPS) * FLOAT(N) )
      NABOVE = IFIX ( (SIGC2 - 1.0 + EPS) * FLOAT(N) )
      IF (NBELOW .GT. 0 .AND. NABOVE .GT. 0) GO TO 9
      IFAIL = 131
      RETURN
    9 IF (NBELOW .GT. MXSTRAY) NBELOW = MXSTRAY
      IF (NABOVE .GT. MXSTRAY) NABOVE = MXSTRAY
C
C        BESTC IS BEST KNOWN CRITERION VALUE OF N-POINT DESIGN:
C        INITIALIZE TO CRITERION VALUE OF CURRENT DESIGN
C
      CALL CRITV (DETPAR, VAR, NREG, AVEVAR, AVESQB, SCALEB,
     &   CRITVAL, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      BESTC = CRITVAL
C
C        PERFORM MAXIMUM OF MAXNSUC SUCCESSFUL EXCURSIONS
C
      DO 10 NSUC = 1, MAXNSUC
C
C        CURRENT DESIGN IS BEST SO FAR - SAVE IN BDESN
C
      DO 400 L = 1, NREG
  400 BDESN(L) = DESN(L)
C
C        NFC IS NUMBER OF FAILED CRITERION VALUES.  IABOVE AND IBELOW
C        ARE FLAGS FOR EXCURSION DEVIATION OF NABOVE OR -NBELOW
C
      NFC = 0
      IABOVE = 0
      IBELOW = 0
C
C        NO MORE THAN MAXFAIL FAILED EXCURSIONS ARE ALLOWED IN
C        SEARCHING FOR A SUCCESSFUL EXCURSION
C
      DO 20 NFAIL = 1, MAXFAIL
C
C        IF NFAIL > 1 LAST EXCURSION WAS FAILURE SO REPLACE
C        CURRENT DESIGN AND WORKING ARRAYS BY THOSE OF BEST SO FAR
C        IF CURRENT DESIGN INFERIOR
C
      IF (NFAIL .EQ. 1 .OR. CRITVAL / BESTC .LT. 1.0 + EPS) GO TO 21
      DO 402 L = 1, NREG
  402 DESN(L) = BDESN(L)
      CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, ALPHA, N, ICRIT, 0, 1, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        CHOOSE TO ADD (IAS = 1) OR SUBTRACT (IAS = -1) FIRST POINT OF
C        EXCURSION.  IF NFAIL ODD THEN ADD, IF EVEN THEN SUBTRACT.
C        IF EXCURSION DEVIATION OF NABOVE OR -NBELOW HAS BEEN REACHED
C        THEN ALWAYS MOVE IN OPPOSITE DIRECTION.
C
```

```
   21 IAS = -1
      IF (NFAIL - (NFAIL / 2) * 2 .EQ. 1) IAS = 1
      IF (IBELOW .EQ. 1) IAS = 1
      IF (IABOVE .EQ. 1) IAS = -1
      CALL ADDSUB (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, IAS, NCUR, N, ALPHA, NUPD, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        EXCURSION LENGTH (NUMBER OF ADDITIONS AND SUBTRACTIONS OF
C        A POINT LIMITED TO MAXLENG
C
      DO 30 LENGTH = 2, MAXLENG
C
C        SEARCH FC, LIST OF FAILED CRITERION VALUES, FOR CRITVAL
C
      CALL CRITV (DETPAR, VAR, NREG, AVEVAR, AVESQB, SCALEB,
     &   CRITVAL, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      IF (NFC .EQ. 0) GO TO 35
      DO 31 I = 1, NFC
      R = CRITVAL / FC(I)
      IF (R .LT. 1.0 - EPS .OR. R .GT. 1.0 + EPS) GO TO 31
C
C        ON LIST - MOVE AWAY FROM N POINTS
C
      IF (NCUR .LT. N + NABOVE
     &   .AND. NCUR .GT. N - NBELOW) GO TO 32
C
C        DEVIATING TOO FAR: EXCURSION FAILS.  CHANGE DEVIATION FLAG AND
C        REPLACE DESN BY BEST DESIGN SO FAR.
C
      IF (NCUR .GT. N) IABOVE = 1
      IF (NCUR .LT. N) IBELOW = 1
      DO 404 L = 1, NREG
  404 DESN(L) = BDESN(L)
C
C        IF DEVIATIONS OF NABOVE AND -NBELOW HAVE OCCURED ABANDON
C        FURTHER SEARCHING AND RETURN
C
      IF (IABOVE .EQ. 1 .AND. IBELOW .EQ. 1) RETURN
C
C        EXCESSIVE DEVIATION HAS ONLY OCCURED IN ONE DIRECTION SO
C        REPLACE WORKING ARRAYS AND TRY ANOTHER EXCURSION
C
      CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, ALPHA, N, ICRIT, 0, 1, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      NCUR = N
      GO TO 20
C
C        DEVIATION NOT EXCESSIVE: MOVE AWAY FROM N POINTS
C
   32 IF (NCUR .GT. N) IAS = 1
      IF (NCUR .LT. N) IAS = -1
      CALL ADDSUB (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, IAS, NCUR, N, ALPHA, NUPD, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      GO TO 30
C
   31 CONTINUE
C
C        NOT ON LIST - IF SUFFICIENT SPACE PUT CRITVAL ON
C        LIST AND MOVE TOWARDS N POINTS
C
   35 NFC = NFC + 1
      IF (NFC .LE. MAXNFC) GO TO 36
C
C        INSUFFICIENT SPACE: REPLACE DESN BY BEST SO FAR AND RETURN
C
      DO 408 L = 1, NREG
  408 DESN(L) = BDESN(L)
      RETURN
```

```
C
C        SUFFICIENT SPACE
C
   36 FC(NFC) = CRITVAL
      IF (NCUR .GT. N) IAS = -1
      IF (NCUR .LT. N) IAS = 1
      CALL ADDSUB (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, IAS, NCUR, N, ALPHA, NUPD, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        IF NOT RETURNED TO N-POINT DESIGN GO TO NEXT STEP OF EXCURSION
C
      IF (NCUR .NE. N) GO TO 30
C
C        N-POINT DESIGN - IF NO IMPROVEMENT ON BEST SO FAR THIS EXCURSION
C        HAS FAILED
C
      CALL CRITV (DETPAR, VAR, NREG, AVEVAR, AVESQB, SCALEB,
     &   CRITVAL, ICRIT, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      IF (CRITVAL / BESTC .GT. 1.0 - EPS) GO TO 20
C
C        SUCCESSFUL EXCURSION: REPLACE BESTC BY CRITVAL
C
      BESTC = CRITVAL
      GO TO 10
C
   30 CONTINUE
C
C        LENGTH OF EXCURSION EXCEEDS MAXLENG - REPLACE CURRENT DESIGN
C        BY BEST AND RETURN.
C
      DO 410 L = 1, NREG
  410 DESN(L) = BDESN(L)
       RETURN
C
   20 CONTINUE
C
C        NUMBER OF FAILURES EXCEEDS MAXFAIL - REPLACE CURRENT DESIGN BY BEST
C        SO FAR AND RETURN.
C
      DO 412 L = 1, NREG
  412 DESN(L) = BDESN(L)
      RETURN
C
   10 CONTINUE
C
C        MAXNSUC SUCCESSFUL ITERATIONS
C
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE ADDSUB (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, IAS, NCUR, N, ALPHA, NUPD, ICRIT, IFAIL)
C
C        ADD OR SUBTRACT A POINT SUCH THAT CRITERION IS MINIMIZED
C
C        IAS      1   ADD A POINT
C                 -1   SUBTRACT A POINT
C        NCUR     NUMBER OF POINTS IN CURRENT DESIGN
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 201   CONSTRAINTS DO NOT ALLOW THE ADDITION OR SUBTRACTION
C                 (ANY OTHER NONZERO VALUE MEANS FAILURE IN A SUBROUTINE
C                 CALLED FROM ADDSUB)
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K), DESN(NREG),
     &   COVPAR(MAXK, K), VAR(NREG), CUC(MAXK, K), FCUCF(NREG),
     &   F(MAXK), CF(MAXK), INDEX(MAXNREG)
C
      COMMON /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &   QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
      RN = 1.0 / FLOAT(N)
      ICH = 0
      BEST = BIG
      ASN = FLOAT(IAS) * FLOAT(N)
```

```
C
      IF (ICRIT .NE. 2HDO) GO TO 20
C
C        ICRIT = 2HDO
C
      IF (IAS .EQ. 1) GO TO 10
      VARMIN = BIG
      DO 11 L = 1, NREG
      IF (VAR(L) .GT. VARMIN .OR.
     &   DESN(L) - CONS(1, L) .LT. RN - EPS) GO TO 11
      VARMIN = VAR(L)
      ICH = L
   11 CONTINUE
      IF (ICH .NE. 0) GO TO 70
      IFAIL = 201
      RETURN
   10 VARMAX = 0.0
      DO 13 L = 1, NREG
      IF (VAR(L) .LT. VARMAX .OR.
     &   CONS(2, L) - DESN(L) .LT. RN - EPS) GO TO 13
      VARMAX = VAR(L)
      ICH = L
   13 CONTINUE
      IF (ICH .NE. 0) GO TO 70
      IFAIL = 201
      RETURN
C
   20 IF (ICRIT .NE. 2HAV) GO TO 30
C
C        ICRIT = 2HAV
C
      DO 21 L = 1, NREG
      IF (IAS .EQ. -1 .AND. DESN(L) - CONS(1, L) .LT. RN - EPS)
     &   GO TO 21
      IF (IAS .EQ. 1 .AND. CONS(2, L) - DESN(L) .LT. RN - EPS)
     &   GO TO 21
      DEL = - FCUCF(L) / (ASN + VAR(L))
      IF (DEL .GT. BEST) GO TO 21
      ICH = L
      BEST = DEL
   21 CONTINUE
      IF (ICH .NE. 0) GO TO 70
      IFAIL = 201
      RETURN
C
   30 IF (ICRIT .NE. 2HMV) GO TO 50
C
C        ICRIT = 2HMV
C
C        COMPUTE INDEX: CANDIDATE NUMBERS IN DESCENDING ORDER OF VAR
C
      CALL ORDER (VAR, NREG, INDEX, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
      DO 33 L = 1, NREG
      IF (IAS .EQ. -1 .AND. DESN(L) - CONS(1, L) .LT. RN - EPS)
     &   GO TO 33
      IF (IAS .EQ. 1 .AND. CONS(2, L) - DESN(L) .LT. RN - EPS)
     &   GO TO 33
      U = ASN + VAR(L)
      IF (IAS .EQ. -1) V = 1.0 - VAR(L) * RN
      DO 37 J = 1, K
   37 F(J) = FREG(L, J)
      CALL MATVEC (COVPAR, F, MAXK, K, CF)
C
C        COMPUTE AMAXL - MAXIMUM VARIANCE AFTER CHANGE AT L
C
      AMAXL = 0.0
      DO 39 LLX = 1, NREG
      LX = INDEX(LLX)
C
C        COMPUTE BOUND ON VARIANCE AT LX AFTER CHANGE
C
      IF (IAS .EQ. 1) BNDUP = VAR(LX)
      IF (IAS .EQ. -1) BNDUP = VAR(LX) / V
      IF (BNDUP .LT. AMAXL) GO TO 40
```

```
C
C        COMPUTE VARNEW: VARIANCE AT LX AFTER CHANGE
C
      COV = 0.0
      DO 41 J = 1, K
   41 COV = COV + FREG(LX, J) * CF(J)
      VARNEW = VAR(LX) - COV * COV / U
C
      IF (VARNEW .GT. AMAXL) AMAXL = VARNEW
   39 CONTINUE
C
   40 IF (AMAXL .GT. BEST) GO TO 33
      BEST = AMAXL
      ICH = L
   33 CONTINUE
      IF (ICH .NE. 0) GO TO 70
      IFAIL = 201
      RETURN
C
   50 IF (ICRIT .NE. 2HAM) GO TO 70
C
C        ICRIT = 2HAM
C
      DO 51 L = 1, NREG
      IF (IAS .EQ. -1 .AND. DESN(L) - CONS(1, L) .LT. RN - EPS)
     &   GO TO 51
      IF (IAS .EQ.  1 .AND. CONS(2, L) - DESN(L) .LT. RN - EPS)
     &   GO TO 51
      DEL = (1.0 + 2.0 * DESN(L) * ASN + FQQF(L)) * FCUCF(L)
     &   / (ASN + VAR(L)) ** 2
     &   - 2.0 * FQQUCF(L) / (ASN + VAR(L))
      DEL = DEL * SCALEB
C
C        CHANGE IN AVEVAR NORMALIZED FOR NCUR + IAS POINTS
C
      DEL = DEL - (FLOAT(NCUR + IAS) + ALPHA) / FLOAT(N)
     &   * FCUCF(L) / (ASN + VAR(L))
      IF (DEL .GT. BEST) GO TO 51
      ICH = L
      BEST = DEL
   51 CONTINUE
      IF (ICH .NE. 0) GO TO 70
      IFAIL = 201
      RETURN
C
C        UPDATE FOR CHANGE AT CANDIDATE ICH
C
   70 CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, ICH, FLOAT(IAS) / FLOAT(N), ICRIT, 1, 0,
     &   NUPD, IFAIL)
      NCUR = NCUR + IAS
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE ORDER (VAR, NREG, INDEX, IFAIL)
C
C        INDEX   ON EXIT ORDERS VAR I.E. INDEX(1) CONTAINS CANDIDATE
C                NUMBER WITH LARGEST VAR ETC.
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                211   NAG ROUTINE M01ABF FAILS
C
      DIMENSION VAR(NREG), INDEX(NREG), IRANKS(MAXNREG)
C
C
C        NAG ROUTINE M01ABF RANKS VAR IN IRANKS
C
      NAGFAIL = 1
      CALL M01ABF (VAR, 1, NREG, IRANKS, INDEX, NAGFAIL)
      IF (NAGFAIL .NE. 0) IFAIL = 211
      IF (IFAIL .NE. 0) RETURN
C
C        CONVERT RANKS TO INDEX
C
      DO 31 L = 1, NREG
      II = IRANKS(L)
   31 INDEX(II) = L
      RETURN
      END
```

```
      SUBROUTINE QUAD (F, S, IDIM, K, FSF)
C
C        COMPUTE FSF = F TRANSPOSE * S * F WHERE F IS K X 1 AND
C        S IS K X K SYMMETRIC WITH LOWER TRIANGULAR ENTRIES
C
      DIMENSION S(IDIM, IDIM), F(IDIM)
C
      FSF = 0.0
      IF (K .EQ. 1) GO TO 10
      DO 8 I = 2, K
      SIF = 0.0
      DO 9 J = 1, I - 1
    9 SIF = SIF + S(I, J) * F(J)
    8 FSF = FSF + F(I) * SIF
      FSF = 2.0 * FSF
   10 DO 11 I = 1, K
   11 FSF = FSF + F(I) * F(I) * S(I, 1)
      RETURN
      END
C
C     --------------------
C

      SUBROUTINE CRITV (DETPAR, VAR, NREG, AVEVAR, AVESQB, SCALEB,
     &   CRITVAL, ICRIT, IFAIL)
C
C        RETURN VALUE OF CHOSEN CRITERION
C
C        CRITVAL VALUE OF CRITERION
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                121   CRITERION IS NOT DO, AV, MV OR AM
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION VAR(NREG)
C
      IF (ICRIT .NE. 2HDO) GO TO 3
      CRITVAL = DETPAR
      RETURN
C
    3 IF (ICRIT .NE. 2HAV) GO TO 4
      CRITVAL = AVEVAR
      RETURN
C
    4 IF (ICRIT .NE. 2HMV) GO TO 6
      CRITVAL = 0.0
      DO 5 L = 1, NREG
      IF (VAR(L) .GT. CRITVAL) CRITVAL = VAR(L)
    5 CONTINUE
      RETURN
C
    6 IF (ICRIT .NE. 2HAM) GO TO 8
      CRITVAL = AVESQB * SCALEB + AVEVAR
      RETURN
C
    8 IFAIL = 121
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE LOWBND (CONS, NREG, K, DESN, DETPAR,
     &   VAR, FCUCF, AVEVAR, ICRIT, CRITLOW, G, IFAIL)
C
C        RETURN LOWER BOUND ON OPTIMAL CRITERION VALUE FOR AN APPROXIMATE
C        DESIGN AND GRADIENTS
C
C        CRITLOW LOWER BOUND ON VALUE OF CHOSEN CRITERION FOR AN OPTIMAL
C                DESIGN SATISFYING CONSTRAINTS.
C        G       IGNORING ADDITIVE AND MULTIPLICATIVE CONSTANTS, G(L) IS
C                GRADIENT OF CRITERION AT CURRENT DESIGN DESN IN DIRECTION
C                OF DESIGN WITH 1 AT CANDIDATE L, 0 ELSEWHERE (L = 1, NREG)
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                81   CRITERION IS NOT DO, AV OR AM
C               82   NO DESIGN CAN SATISFY CONSTRAINTS
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), DESN(NREG),
     &   VAR(NREG), FCUCF(NREG), G(NREG),
     &   F(MAXK), IUSED(MAXNREG)
C
      COMMON /AM/   AVESQB, SCALEB, FOQUCF(MAXNREG)
```

```
C
      IF (ICRIT .EQ. 2HDO) GO TO 2
      IF (ICRIT .EQ. 2HAV) GO TO 4
      IF (ICRIT .EQ. 2HAM) GO TO 6
      IFAIL = 81
      RETURN
C
C        COMPUTE GRADIENTS FOR ICRIT = 2HDO
C
    2 DO 3 L = 1, NREG
    3 G(L) = -VAR(L)
      GO TO 30
C
C        COMPUTE GRADIENTS FOR ICRIT = 2HAV
C
    4 DO 5 L = 1, NREG
    5 G(L) = -FCUCF(L)
      GO TO 30
C
C        COMPUTE GRADIENTS FOR ICRIT = 2HAM
C
    6 DO 16 L = 1, NREG
   16 G(L) = -FCUCF(L) + 2.0 * SCALEB * ( DESN(L) * FCUCF(L)
     &    - FQQUCF(L) )
C
C        COMPUTE CRITLOW
C
C        FIRST COMPUTE GBND = MINIMUM VALUE OF D(1) * G(1) + ...
C        + D(NREG) * G(NREG) OVER ALL DESIGNS D SATISFYING CONSTRAINTS
C
   30 GBND = 0.0
      ALLOC = 0.0
      DO 31 L = 1, NREG
      ALLOC = ALLOC + CONS(1, L)
      GBND = GBND + G(L) * CONS(1, L)
      IUSED(L) = 0
      IF (CONS(2, L) - CONS(1, L) .LT. EPS) IUSED(L) = 1
   31 CONTINUE
      IF (ALLOC .GT. 1.0 - EPS) GO TO 50
      DO 32 JJJ = 1, NREG
      LPOS = 0
      GMIN = BIG
      DO 33 L = 1, NREG
      IF (IUSED(L) .EQ. 1 .OR. G(L) .GT. GMIN) GO TO 33
      GMIN = G(L)
      LPOS = L
   33 CONTINUE
      IF (LPOS .NE. 0) GO TO 35
      IFAIL = 82
      RETURN
   35 PP = 1.0 - ALLOC
      SLACK = CONS(2, LPOS) - CONS(1, LPOS)
      IF (PP .GT. SLACK) PP = SLACK
      GBND = GBND + G(LPOS) * PP
      ALLOC = ALLOC + PP
      IF (ALLOC .GT. 1.0 - EPS) GO TO 50
      IUSED(LPOS) = 1
   32 CONTINUE
      IFAIL = 82
      RETURN
C
   50 IF (ICRIT .NE. 2HDO) GO TO 52
C
C        COMPUTE CRITLOW FOR ICRIT = 2HDO.  NOTE
C        (1)  LOG DETPAR IS CONVEX WITH RESPECT TO DESIGN
C        (2)  GRADIENTS OF LOG DETPAR ARE G(L) + K SO ADD K TO GBND
C
      GBND = FLOAT(K) + GBND
      CRITLOW = 0.0
      IF (GBND .LT. -100.0) RETURN
      CRITLOW = DETPAR * EXP(GBND)
      RETURN
```

```
C
C       COMPUTE CRITLOW FOR ICRIT = 2HAV OR ICRIT = 2HAM.  NOTE
C       (1)  AVEVAR IS CONVEX WITH RESPECT TO DESIGN
C       (2)  AVESQB IS ASSUMED APPROXIMATELY CONVEX WITH RESPECT TO
C            DESIGN (VALID NEAR OPTIMUM)
C       (3)  GRADIENTS FOR ICRIT = 2HAV AND ICRIT = 2HAM ARE G(L) + AVEVAR
C            SO ADD AVEVAR TO GBND
C
   52 GBND = AVEVAR + GBND
      IF (ICRIT .EQ. 2HAV) CRITLOW = AVEVAR + GBND
      IF (ICRIT .EQ. 2HAM) CRITLOW = AVEVAR + SCALEB * AVESQB + GBND
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE MING (CONS, NREG, DESN, INC, IDEC, G, IFAIL)
C
C        FIND INC AND IDEC
C
C        INC      CANDIDATE MINIMIZING G SUBJECT TO SLACK AT UPPER CONSTRAINT
C        IDEC     CANDIDATE MAXIMIZING G SUBJECT TO SLACK AT LOWER CONSTRAINT
C        G        GRADIENTS (DESCRIBED IN SUBROUTINE LOWBND)
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 91   NO SLACK AT ANY CANDIDATE
C                 92   MINIMUM AND MAXIMUM AT SAME CANDIDATE
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), DESN(NREG), G(NREG)
C
      INC = 0
      IDEC = 0
      GMIN = BIG
      GMAX = -BIG
      DO 3 L = 1, NREG
      IF (G(L) .GT. GMIN .OR. CONS(2, L) - DESN(L) .LT. EPS)
     &   GO TO 2
      INC = L
      GMIN = G(L)
    2 IF (G(L) .LT. GMAX .OR. DESN(L) - CONS(1, L) .LT. EPS)
     &   GO TO 3
      IDEC = L
      GMAX = G(L)
    3 CONTINUE
      IF (INC .EQ. 0 .OR. IDEC .EQ. 0) IFAIL = 91
      IF (INC .EQ. IDEC) IFAIL = 92
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE OPTRAN (CONS, NREG, FREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, AVEVAR, INC, IDEC, BETAOPT, ICRIT, IFAIL)
C
C        FIND OPTIMAL TRANSFER FROM DESN(IDEC) TO DESN(INC).  FOR ICRIT = 2HDO
C        OPTIMUM CALCULATED IN OPTRAN.  FOR ICRIT = 2HAV AND ICRIT = 2HAM
C        CALCULATE QUANTITIES IN COMMON /FN1/ TO ENABLE QUICK CRITERION
C        EVALUATION FOR A GIVEN TRANSFER WHEN OPTIMIZING NUMERICALLY VIA
C        NAG ROUTINE M01ABF.  COMMON /FN1/ COMMUNICATES WITH FUNCT1, THE
C        SUBROUTINE CALLED BY E04ABF FOR CRITERION EVALUATION.
C
C        INC      CANDIDATE AT WHICH DESN IS TO BE INCREASED
C        IDEC     CANDIDATE AT WHICH DESN IS TO BE DECREASED BY SAME AMOUNT
C        BETAOPT  AMOUNT OF TRANFER FROM IDEC TO INC THAT OPTIMIZES
C                 CHOSEN CRITERION SUBJECT TO CONSTRAINTS
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 161   CRITERION IS NOT DO, AV OR AM
C                 162   OPTIMAL TRANSFER LESS THAN EPS
C                 163   PARAMETER OUT OF RANGE WHEN NAG ROUTINE E04ABF CALLED
C                 164   NO CONVERGENCE AFTER 30 ITERATIONS WHEN E04ABF CALLED
C        (OTHER PARAMETERS IN LIST AND COMMON /AM/ DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K), DESN(NREG),
     &   COVPAR(MAXK, K), VAR(NREG), CUC(MAXK, K), FCUCF(NREG),
     &   FI(MAXK), FD(MAXK), CFI(MAXK), W(MAXK),
     &   IWORKSP(3), WORKSP(13)
```

```
C
      COMMON /FN1/  VARI, VARD, COVID, AII, ADD, AID,
     &              TII, TID, TDD, SII, SID, SDD,
     &              IDUM1, DUM2, DUM3, DUM4
     &       /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &              QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
      EXTERNAL FUNCT1
C
      IDUM1 = ICRIT
      DUM2 = AVEVAR
      DUM3 = AVESQB
      DUM4 = SCALEB
      IF (ICRIT .EQ. 2HDO .OR. ICRIT .EQ. 2HAV
     &   .OR. ICRIT .EQ. 2HMV .OR. ICRIT .EQ. 2HAM) GO TO 4
      IFAIL = 161
      RETURN
C
C        BU IS UPPER BOUND ON TRANSFER (FROM CONSTRAINTS)
C
    4 BU = CONS(2, INC) - DESN(INC)
      SLACK = DESN(IDEC) - CONS(1, IDEC)
      IF (SLACK .LT. BU) BU = SLACK
C
      DO 5 J = 1, K
      FD(J) = FREQ(IDEC, J)
    5 FI(J) = FREQ(INC, J)
      CALL MATVEC (COVPAR, FI, MAXK, K, CFI)
      COVID = 0.0
      DO 6 J = 1, K
    6 COVID = COVID + FD(J) * CFI(J)
C
      IF (ICRIT .NE. 2HDO) GO TO 11
C
C        ICRIT = 2HDO SO BETAOPT CAN BE DIRECTLY CALCULATED
C
      BETAOPT = 0.5 * (VAR(INC) - VAR(IDEC)) /
     &   (VAR(INC) * VAR(IDEC) - COVID ** 2)
      IF (BETAOPT .GT. BU) BETAOPT = BU
      IF (BETAOPT .LT. EPS) IFAIL = 162
      RETURN
C
C        ICRIT = 2HAV OR ICRIT = 2HAM
C
   11 VARI = 0.0
      DO 8 J = 1, K
    8 VARI = VARI + FI(J) * CFI(J)
      CALL QUAD (FD, COVPAR, MAXK, K, VARD)
C
      AII = FCUCF(INC)
      ADD = FCUCF(IDEC)
      AID = 0.0
      DO 9 M = 1, K
      DO 9 J = 1, K
    9 AID = AID + FI(M) * CUC(M, J) * FD(J)
C
      IF (ICRIT .NE. 2HAM) GO TO 40
C
C        ICRIT = 2HAM
C
      TII = FQQUCF(INC) - DESN(INC) * FCUCF(INC)
      TDD = FQQUCF(IDEC) - DESN(IDEC) * FCUCF(IDEC)
      TID = 0.0
      DO 25 M = 1, K
      DO 25 J = 1, K
   25 TID = TID + FI(M) * QQUC(M, J) * FD(J)
      TID = TID - (DESN(INC) + DESN(IDEC)) * AID / 2.0
C
      SII = 1.0 + FQQF(INC) - 2.0 * DESN(INC) * VARI
      SDD = 1.0 + FQQF(IDEC) - 2.0 * DESN(IDEC) * VARD
      SID = 0.0
      DO 34 M = 1, K
      DO 34 J = 1, K
   34 SID = SID + FI(M) * QQ(M, J) * FD(J)
      SID = SID - (DESN(INC) + DESN(IDEC)) * COVID
```

```
C
   40 E1 = 0.0
      E2 = 0.0
      BL = 0.0
      MAXCAL = 30
      NAGFAIL = 1
      CALL E04ABF (FUNCT1, E1, E2, BL, BU, MAXCAL,
     &   BETAOPT, OPT, NAGFAIL)
      IF (BETAOPT .LT. EPS) IFAIL = 162
      IF (NAGFAIL .EQ. 1) IFAIL = 163
      IF (NAGFAIL .EQ. 2) IFAIL = 164
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE FUNCT1 (BETA, C)
C
C        CALLED BY NAG ROUTINE E04ABF TO CALCULATE CRITERION VALUE
C
C        BETA     TRANSFER
C        C        CRITERION VALUE FOR TRANSFER
C        (COMMON /FN1/ CALCULATED IN OPTRAN)
C
      COMMON /FN1/  VARI, VARD, COVID, AII, ADD, AID,
     &              TII, TID, TDD, SII, SID, SDD,
     &              ICRIT, AVEVAR, AVESQB, SCALEB
C
      COEFF = BETA / (1.0 + BETA * (VARI - VARD) - BETA ** 2 *
     &   (VARI * VARD - COVID ** 2))
      D11 = 1.0 - BETA * VARD
      D12 = BETA * COVID
      D22 = -(1.0 + BETA * VARI)
C
      IF (ICRIT .NE. 2HAV) GO TO 30
C
C        ICRIT = 2HAV
C
      C = AVEVAR - COEFF * (D11 * AII + 2.0 * D12 * AID + D22 * ADD)
      RETURN
C
C        ICRIT = 2HAM
C
   30 E11 = AII * D11 ** 2 + 2.0 * AID * D11 * D12 + ADD * D12 ** 2
      E22 = ADD * D22 ** 2 + 2.0 * AID * D22 * D12 + AII * D12 ** 2
      E12 = AII * D11 * D12 + AID * (D12 ** 2 + D11 * D22)
     &   + ADD * D22 * D12
      C = AVESQB - 2.0 * COEFF * (D11 * TII
     &   + 2.0 * D12 * TID + D22 * TDD)
     &   + COEFF ** 2 * (E11 * SII + 2.0 * E12 * SID + E22 * SDD)
      C = C * SCALEB + (AVEVAR - COEFF * (D11 * AII
     &   + 2.0 * D12 * AID + D22 * ADD))
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   CUC, FCUCF, AVEVAR, KAN, BETA, ICRIT, IEX, ILC, NUPD, IFAIL)
C
C        UPDATE WORKING ARRAYS AND SCALARS
C
C        KAN      CANDIDATE AT WHICH CHANGE TAKES PLACE
C        BETA     AMOUNT OF CHANGE
C        IEX      1    VAR ALWAYS UPDATED (EXCURSION ALGORITHM)
C                 0    ONLY COMPUTE VAR IF NECESSARY FOR CRITERION
C        ILC      0    UPDATE DESN FOR CHANGE AT DESN(KAN)
C                 1    DO NOT UPDATE DESN (ONLY SOME CALLS FROM BANDB)
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 171    BETA LESS THAN EPS
C                 172    CHANGE VIOLATES CONSTRAINTS
C                 173    CRITERION IS NOT DO, AV, MV OR AM
```

```
C       (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
C       CRITERION   NECESSARY TO UPDATE
C
C       DO          COVPAR, DETPAR, VAR
C       AV          COVPAR, AVEVAR, CUC, FCUCF
C       MV          COVPAR, VAR (IEX = 1 ALWAYS FOR MV)
C       AM          COVPAR, AVEVAR, CUC, FCUCF, AVESQB, QQ, QQUC,
C                   FQQF, FQQUCF
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K),
     &   DESN(NREG), COVPAR(MAXK, K), VAR(NREG), CUC(MAXK, K),
     &   FCUCF(NREG), F(MAXK), CF(MAXK), CFBAR(MAXK), CUCF(MAXK),
     &   QQF(MAXK), QQUCF(MAXK), V(MAXK), W(MAXK)
C
      COMMON /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &                QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
      NUPD = NUPD + 1
C
      IF (ABS(BETA) .GT. EPS) GO TO 3
      IFAIL = 171
      RETURN
C
    3 IF (ICRIT .EQ. 2HDO .OR. ICRIT. EQ. 2HAV
     &   .OR. ICRIT .EQ. 2HMV .OR. ICRIT .EQ. 2HAM) GO TO 5
      IFAIL = 173
      RETURN
C
C
C       COMPUTE SOME QUANTITIES NECESSARY FOR UPDATING
C
    5 DO 6 J = 1, K
    6 F(J) = FREG(KAN, J)
      CALL MATVEC (COVPAR, F, MAXK, K, CF)
      VARKAN = 0.0
      DO 7 J = 1, K
    7 VARKAN = VARKAN + F(J) * CF(J)
      DENOM = 1.0 / BETA + VARKAN
      DO 9 J = 1, K
    9 CFBAR(J) = CF(J) / DENOM
      IF (ICRIT .NE. 2HAV .AND. ICRIT .NE. 2HAM) GO TO 15
      CALL MATVEC (CUC, F, MAXK, K, CUCF)
      AKAN = FCUCF(KAN)
      IF (ICRIT .NE. 2HAM) GO TO 15
      CALL MATVEC (QQ, F, MAXK, K, QQF)
      CALL MATVEC (QQUC, F, MAXK, K, QQUCF)
      BKAN = 1.0 + 2.0 * DESN(KAN) / BETA + FQQF(KAN)
      CKAN = FQQUCF(KAN)
      DO 10 J = 1, K
      V(J) = QQF(J) - BKAN * CFBAR(J)
   10 W(J) = CUCF(J) - AKAN * CFBAR(J)
C
C       UPDATE CRITERION VALUES
C
   15 IF (ICRIT .EQ. 2HDO) DETPAR = DETPAR / (1.0 + BETA * VARKAN)
      IF (ICRIT .EQ. 2HAV .OR. ICRIT .EQ. 2HAM)
     &   AVEVAR = AVEVAR - AKAN / DENOM
      IF (ICRIT .EQ. 2HAM) AVESQB = AVESQB - 2.0 * CKAN / DENOM
     &   + BKAN * AKAN / DENOM ** 2
C
C       UPDATE MATRICES
C
      DO 17 I = 1, K
      DO 17 J = 1, I
      COVPAR(I, J) = COVPAR(I, J) - CF(I) * CFBAR(J)
   17 COVPAR(J, I) = COVPAR(I, J)
      IF (ICRIT .NE. 2HAV .AND. ICRIT .NE. 2HAM) GO TO 30
      DO 18 I = 1, K
      DO 18 J = 1, I
      CUC(I, J) = CUC(I, J) - CFBAR(I) * CUCF(J)
     &   - CUCF(I) * CFBAR(J) + AKAN * CFBAR(I) * CFBAR(J)
   18 CUC(J, I) = CUC(I, J)
      IF (ICRIT .NE. 2HAM) GO TO 30
      DO 20 I = 1, K
      DO 20 J = 1, I
      QQ(I, J) = QQ(I, J) - CFBAR(I) * QQF(J)
     &   - QQF(I) * CFBAR(J) + BKAN * CFBAR(I) * CFBAR(J)
      QQ(J, I) = QQ(I, J)
```

```
      QQUC(I, J) = QQUC(I, J) - CFBAR(I) * QQUCF(J)
     &   - QQUCF(I) * CFBAR(J) + CKAN * CFBAR(I) * CFBAR(J)
     &   - 0.5 * (V(I) * W(J) + W(I) * V(J)) / DENOM
   20 QQUC(J, I) = QQUC(I, J)
C
C        UPDATE QUADRATIC FORMS
C
   30 DO 40 L = 1, NREG
      DLK = 0.0
      DO 41 J = 1, K
   41 DLK = DLK + FREG(L, J) * CF(J)
      DLKBAR = DLK / DENOM
      IF (ICRIT .EQ. 2HDO .OR. IEX .EQ. 1) VAR(L) =
     &   VAR(L) - DLKBAR * DLK
      IF (ICRIT .NE. 2HAV .AND. ICRIT .NE. 2HAM) GO TO 40
      ALK = 0.0
      DO 43 J = 1, K
   43 ALK = ALK + FREG(L, J) * CUCF(J)
      FCUCF(L) = FCUCF(L) - 2.0 * ALK * DLKBAR + AKAN * DLKBAR ** 2
      IF (ICRIT .NE. 2HAM) GO TO 40
      BLK = 0.0
      CLK = 0.0
      DO 44 J = 1, K
      BLK = BLK + FREG(L, J) * QQF(J)
   44 CLK = CLK + FREG(L, J) * QQUCF(J)
      FQQF(L) = FQQF(L) - 2.0 * BLK * DLKBAR + BKAN * DLKBAR ** 2
      FQQUCF(L) = FQQUCF(L) - 2.0 * CLK * DLKBAR + CKAN * DLKBAR ** 2
     &   - (BLK - BKAN * DLKBAR) * (ALK - AKAN * DLKBAR) / DENOM
   40 CONTINUE
C
      IF (ILC .EQ. 1) RETURN
C
C        UPDATE DESN
C
      DESN(KAN) = DESN(KAN) + BETA
      IF (DESN(KAN) .LT. CONS(1, KAN) - EPS .OR.
     &   DESN(KAN) .GT. CONS(2, KAN) + EPS) IFAIL = 172
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE PREPOST (A, W, K)
C
C        COMPUTE A = W * A * TRANSPOSE OF W WHERE A IS SYMMETRIC
C
      DIMENSION A(MAXK, *), W(MAXK, K), WORKSP(MAXK)
C
      DO 12 J = 1, K
      DO 13 I = 1, K
   13 WORKSP(I) = A(I, J)
      DO 12 I = 1, K
      A(I, J) = 0.0
      DO 12 M = 1, K
   12 A(I, J) = A(I, J) + W(I, M) * WORKSP(M)
C
      DO 15 I = 1, K
      DO 16 J = 1, K
   16 WORKSP(J) = A(I, J)
      DO 15 J = 1, I
      A(I, J) = 0.0
      DO 17 M = 1, K
   17 A(I, J) = A(I, J) + WORKSP(M) * W(J, M)
   15 A(J, I) = A(I, J)
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE OUTPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   CUC, FCUCF, II, NDES, ICRIT, IMOD, IALG, GAMMA, RATIO,
     &   ITERS, N, IFAIL)
C
C        OUTPUT DESIGN PROPERIES FOR DESN
C
```

```
C        II      DESIGN REFERENCE NUMBER
C        ITERS   NUMBER OF ITERATIONS TO FIND DESN: FOR APPROXIMATE
C                ALGORITHM NUMBER OF TRANSFERS; FOR EXCURSION ALGORITHM
C                NUMBER OF TIMES POINT ADDED OR SUBTRACTED; AND FOR
C                BRANCH AND BOUND ALGORITHM NODE AT WHICH DESN FOUND
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., NONZERO VALUE MEANS FAILURE
C                IN A SUBROUTINE CALLED FROM OUTPROP
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION FREG(MAXNREG, K), DESN(NREG), COVPAR(MAXK, K),
     &   VAR(NREG), CUC(MAXK, K), FCUCF(NREG)
C
      COMMON /AM/   AVESQB, SCALEB
C
C        COMPUTE DESIGN PROPERTIES (WITH ZERO ADJUSTMENT)
C
      CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, CUC, FCUCF, AVEVAR, 0.0, N, ICRIT, 1, 0, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      AMAXVAR = 0.0
      DO 10 L = 1, NREG
      IF (VAR(L) .GT. AMAXVAR) AMAXVAR = VAR(L)
   10 CONTINUE
      IF (ICRIT .EQ. 2HAM) AVEMSE = AVEVAR + SCALEB * AVESQB
C
C        OUTPUT PROBLEM SUMMARY IF FIRST DESIGN
C
      IF (II .EQ. 1) WRITE (2, 102) NREG, IMOD, ICRIT, N, K, IALG
  102 FORMAT (// 1X, 'CANDIDATES', I4, 10X, 'MODEL', 8X, A1,
     &   10X, 'CRITERION',3X, A2 /
     &   1X, 'OBS', 7X, I4, 10X, 'PARAMETERS', I4,
     &   10X, 'ALGORITHM', 3X, A2)
      IF (II .EQ. 1 .AND. ICRIT .EQ. 2HAM) WRITE (2, 101) RATIO
  101 FORMAT (1X, 'RATIO', F9.3)
      IF (II .EQ. 1 .AND. IALG .EQ.2HBR) WRITE(2, 109) GAMMA
  109 FORMAT (1X, 'GAMMA', F9.3)
C
C        WRITE HEADINGS IF FIRST DESIGN
C
      IF(II .EQ. 1 .AND. ICRIT .NE. 2HAM) WRITE (2, 103)
  103 FORMAT (/ 1X, 55(1H-) / 21X, 'DETERMINANT' /
     &   ' DESIGN', 15X, 'COVARIANCE', 5X, 'MAXIMUM',
     &   5X, 'AVERAGE' /
     &   4X, 'REF  ITERATIONS', 7X, 'MATRIX', 2(4X, 'VARIANCE') /
     &   1X, 55(1H-) /)
      IF (II .EQ. 1 .AND. ICRIT .EQ. 2HAM) WRITE (2, 104)
  104 FORMAT (/ 1X, 79(1H-) / 21X, 'DETERMINANT', 28X, 'UNSCALED' /
     &   ' DESIGN', 15X, 'COVARIANCE', 5X, 'MAXIMUM',
     &   3(5X, 'AVERAGE') /
     &   4X, 'REF  ITERATIONS', 7X, 'MATRIX', 2(4X, 'VARIANCE'),
     &   5X, 'SQ BIAS', 9X, 'MSE' / 1X, 79(1H-) /)
C
C        WRITE DESIGN PROPERTIES
C
      IF (ICRIT .NE. 2HAM) WRITE (2, 100) II, ITERS, DETPAR,
     &   AMAXVAR, AVEVAR
  100 FORMAT (I7, I12, E13.4, 4E12.4)
      IF (ICRIT .EQ. 2HAM) WRITE (2, 100) II, ITERS, DETPAR,
     &   AMAXVAR, AVEVAR, AVESQB, AVEMSE
C
C        OUTPUT NORMALIZATION MESSAGE IF LAST DESIGN
C
      IF (II .EQ. NDES) WRITE (2,107)
  107 FORMAT (/ ' ALL PROPERTIES NORMALIZED FOR NUMBER OF',
     &   ' OBSERVATIONS AND ERROR VARIANCE' )
      RETURN
      END
C
C     ------------------
C
      SUBROUTINE OUTDES (NCOLS, II, DESN, NREG, N, IFAIL)
C
C        OUTPUT II AND DESN
C
C        NCOLS   PAGE WIDTH AVAILABLE
C        II      DESIGN REFERENCE NUMBER
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                71   INSUFFICIENT PAGE WIDTH TO OUTPUT DESIGN
C                72   CANDIDATE WITH REPLICATION OF 100 OR MORE
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
```

```
C
      DIMENSION DESN(NREG), IPOS(MAXNREG), REP(MAXNREG)
C
      IF (NCOLS - 7 .GE. 11) GO TO 3
      IFAIL = 71
      RETURN
C
C        WRITE HEADINGS IF FIRST DESIGN
C
    3 IF (II .EQ. 1) WRITE (2, 115) NCOLS - 1, NCOLS - 1
  115 FORMAT (// 1X, =(1H-) / ' DESIGN' / 4X, 'REF', 3X,
     &   'CANDIDATES WITH NONZERO REPLICATIONS' / 1X, =(1H-) /)
C
C        DETERMINE CANDIDATES WITH POSITIVE REPLICATIONS AND WIDTH REQUIRED
C        TO OUTPUT EACH CANDIDATE AND REPLICATION
C
      NPOS = 0
      KOLSREQ = 4
      DO 4 L = 1, NREG
      IF (DESN(L) .LT. EPS) GO TO 4
      AMN = DESN(L) * FLOAT(N)
      IF (AMN .LT. 100.0) GO TO 5
      IFAIL = 72
      RETURN
    5 NPOS = NPOS + 1
      IPOS(NPOS) = L
      REP(NPOS) = AMN
      IF (KOLSREQ .LT. 8 .AND. IFIX(AMN + EPS) .NE. 1) KOLSREQ = 8
      FRACT = AMN - AINT(AMN)
      IF (KOLSREQ .LT. 11 .AND. FRACT .GT. EPS
     &   .AND. FRACT .LT. 1.0 - EPS) KOLSREQ = 11
    4 CONTINUE
C
C        OUTPUT CANDIDATES WITH POSITIVE REPLICATIONS AND THEIR REPLICATIONS
C
      NONLINE = (NCOLS - 7) / KOLSREQ
      NLINE = (NPOS + NONLINE - 1) / NONLINE
      DO 8 LINE = 1, NLINE
      NDONE = (LINE - 1) * NONLINE
      IF (LINE .EQ. NLINE) NONLINE = NPOS - (NLINE - 1) * NONLINE
      IF (LINE .NE. 1) GO TO 7
      IF (KOLSREQ .EQ. 4) WRITE (2, 100) II, NONLINE,
     &   (IPOS(L), L = 1, NONLINE)
  100 FORMAT (I5, 2X, =I4)
      IF (KOLSREQ .EQ. 8) WRITE (2, 101) II, NONLINE,
     &   ((IPOS(L), IFIX(REP(L) + EPS)), L = 1, NONLINE)
  101 FORMAT (I5, 2X, =(I4, '(', I2, ')' ))
      IF (KOLSREQ .EQ. 11) WRITE (2, 102) II, NONLINE,
     &   ((IPOS(L), REP(L)), L = 1, NONLINE)
  102 FORMAT (I5, 2X, =(I4, '(', F5.2, ')' ))
      GO TO 8
    7 IF (KOLSREQ .EQ. 4) WRITE (2, 103) NONLINE,
     &   (IPOS(L), L = NDONE + 1, NDONE + NONLINE)
  103 FORMAT (7X, =I4)
      IF (KOLSREQ .EQ. 8) WRITE (2, 104) NONLINE,
     &   ((IPOS(L), IFIX(REP(L) + EPS)),
     &   L = NDONE + 1, NDONE + NONLINE)
  104 FORMAT (7X, =(I4, '(', I2, ')' ))
      IF (KOLSREQ .EQ. 11) WRITE (2, 105) NONLINE,
     &   ((IPOS(L), REP(L)), L = NDONE + 1, NDONE + NONLINE)
  105 FORMAT (7X, =(I4, '(', F5.2, ')' ))
    8 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MS7MAN (II, DESN, NREG, IWR)
C
C        MS7MAN MANAGES A LIST OF DESIGNS ON RANDOM ACCESS DEVICE TAPE7
C        IF IWR = 1 WRITE DESN IN POSITION II
C        IF IWR = -1 READ DESN FROM POSITION II
C
      DIMENSION DESN(NREG), INDEX(MAXNDP1)
C
      DATA IOPEN /0/
C
      IF (IOPEN .EQ. 1) GO TO 3
      CALL OPENMS (7, INDEX, MAXNDP1, 0)
      IOPEN = 1
    3 IF (IWR .EQ. 1) CALL WRITMS (7, DESN, NREG, II, -1, 0)
      IF (IWR .EQ. -1) CALL READMS (7, DESN, NREG, II)
      RETURN
      END
C
C     ----------------------
C
      SUBROUTINE FURPROP (FREG, NREG, K, DESN, COVPAR, VAR,
     &   ICRIT, NDES, INTER, IFAIL)
C
C        OUTPUT FURTHER PROPERTIES OF SPECIFIED DESIGNS
C
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., NONZERO VALUE MEANS FAILURE
C                IN A SUBROUTINE CALLED FROM FURPROP
C        (OTHER PARAMETERS IN LIST AND COMMON /AM/ DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION FREG(MAXNREG, K), DESN(NREG), COVPAR(MAXK, K),
     &   VAR(NREG), IOPS(10), F(MAXK)
C
      COMMON /AM/   AVESQB, SCALEB
C
C        COMMON /SPACE1/ USED ONLY TO REDUCE STORAGE, NOT FOR COMMUNICATION
C
      COMMON /SPACE1/ XTX(MKP1, MAXK)
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
C        READ DESIGN REF FOR FURTHER PROPERTIES
C
  300 DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ ' FURTHER PROPERIES OF ANY DISPLAYED DESIGN?',
     &   '  ENTER A DESIGN REF NUMBER' / ' FOR REQUIRED DESIGN',
     &   ' OR 0 (ZERO) FOR NO FURTHER PROPERTIES' /)
      READ (1, *) IREF
      IF (EOF(1) .EQ. 0.0) GO TO 2
      WRITE (2, 120)
  120 FORMAT (/ ' ERROR: END OF FILE WHEN READING DESIGN REF',
     &   ' FOR FURTHER PROPERTIES' )
      GO TO 4
    2 IF (IREF .EQ. 0) RETURN
      IF (IREF .GE. 1 .AND. IREF .LE. NDES) GO TO 20
      WRITE (2, 121) NDES
  121 FORMAT (/ ' ERROR: DESIGN REF FOR FURTHER PROPERTIES LESS',
     &   ' THAN 1 OR EXCEEDS', I5)
    4 CONTINUE
      STOP
C
C         READ OPTIONS (ONLY FIRST 10 CHARACTERS READ)
C
   20 DO 5 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 101)
  101 FORMAT (/ ' OPTIONS?  ENTER COMBINATION OF C V M OR HELP' /)
      READ (1, 102) IOPS
  102 FORMAT (10A1)
      IF (EOF(1) .EQ. 0.0) GO TO 9
      WRITE (2, 125)
  125 FORMAT (/ ' ERROR: END OF FILE WHEN READING OPTIONS FOR',
     &   ' FURTHER PROPERTIES' )
      GO TO 7
    9 IC = 0
      IV = 0
      IM = 0
      DO 6 I = 1, 10
      IF (IOPS(I) .EQ. 1HH) GO TO 7
      IF (IOPS(I) .NE. 1HC .AND. IOPS(I) .NE. 1HV
     &   .AND. IOPS(I) .NE. 1HM .AND. IOPS(I) .NE. 1H ) GO TO 10
```

```
      IF (IOPS(I) .EQ. 1HC) IC = 1
      IF (IOPS(I) .EQ. 1HV) IV = 1
      IF (IOPS(I) .EQ. 1HM .AND. ICRIT .EQ. 2HAM) IM = 1
    6 CONTINUE
      IF (IM .EQ. 1) IV = 0
      IF (IC .EQ. 1 .OR. IV .EQ. 1 .OR. IM .EQ. 1) GO TO 11
   10 WRITE (2, 128)
  128 FORMAT (/ ' OPTION REQUEST FOR FURTHER PROPERTIES IS',
     &    ' NOT C V OR M' )
    7 WRITE (2, 105)
  105 FORMAT (/ ' OPTION' //
     &    ' C', 7X, 'COVARIANCE MATRIX OF PARAMETER ESTIMATORS' /
     &    ' V', 7X, 'VARIANCES OF RESPONSE ESTIMATORS' /
     &    ' M', 7X, 'VARIANCES, SQUARE BIASES (AVERAGED OVER',
     &    ' POTENTIAL BAD DEPARTURES)' / 9X, 'AND MEAN SQUARE',
     &    ' ERRORS (AVERAGED ...) OF RESPONSE ESTIMATORS' /
     &    9X, '(ONLY AVAILABLE WITH AM CRITERION)' )
    5 CONTINUE
      STOP
C
   11 WRITE (2, 107) IREF
  107 FORMAT (// ' DESIGN REF', I7 / 1X, 17(1H-) )
C
C        GET DESIGN IREF UNLESS NDES = 1 IN WHICH CASE
C        ONLY ONE DESIGN IS AVAILABLE AND ALREADY IN DESN
C
      IF (NDES .GE. 2) CALL MS7MAN (IREF, DESN, NREG, -1)
C
C        COMPUTE COVPAR
C
      CALL INFMAT (FREG, NREG, K, DESN, XTX)
      CALL INFINV (XTX, COVPAR, K, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C         WRITE COVPAR IF IC = 1
C
      IF (IC .NE. 1) GO TO 40
      WRITE (2, 108)
  108 FORMAT (/ ' COVARIANCE MATRIX OF PARAMETER ESTIMATORS' /)
      CALL WSMAT (COVPAR, MAXK, K, 6)
C
C        COMPUTE VARIANCES IF IV = 1 OR IM = 1
C
   40 IF (IV .NE. 1 .AND. IM .NE. 1) GO TO 1
      DO 41 L = 1, NREG
      DO 42 J = 1, K
   42 F(J) = FREG(L, J)
   41 CALL QUAD (F, COVPAR, MAXK, K, VAR(L))
C
C        WRITE VARIANCES IF IV = 1
C
      IF (IV .NE. 1) GO TO 60
      NONLINE = 4
      IF (NREG .LT. 4) NONLINE = NREG
      WRITE(2, 110) NONLINE - 1
  110 FORMAT (// ' CAND', 3X, 'VARIANCE', =(4X, 'CAND', 3X,
     &    'VARIANCE') /)
      WRITE (2, 112) ( (L, VAR(L)), L = 1, NREG)
  112 FORMAT ( (I5, E11.4, 3(I8, E11.4)) )
C
C        IF IM = 1 COMPUTE SQUARED BIASES AND MEAN SQUARE ERRORS
C        THEN WRITE THEM WITH VARIANCES
C
   60 IF (IM .NE. 1) GO TO 1
      WRITE (2, 115)
  115 FORMAT (/ 21X, 'UNSCALED' / 1X, 'CAND',
     &    4X, 'VARIANCE', 5X, 'SQ BIAS', 9X, 'MSE' /)
      DO 62 L = 1, NREG
      CALL SQBIAS (L, FREG, NREG, K, DESN, COVPAR,
     &    DESN(L) * VAR(L), SQB)
      FMSE = VAR(L) + SCALEB * SQB
   62 WRITE (2, 117) L, VAR(L), SQB, FMSE
  117 FORMAT (I5, 3E12.4)
C
    1 WRITE (2, 150)
  150 FORMAT (/ ' ALL PROPERTIES NORMALIZED FOR NUMBER OF',
     &    ' OBSERVATIONS AND ERROR VARIANCE' )
      GO TO 300
C
      END
```

```
      SUBROUTINE SQBIAS (KAN, FREG, NREG, K, DESN, COVPAR, G, SQB)
C
C        COMPUTE SQUARED BIAS FOR CANDIDATE KAN
C
C        KAN      CANDIDATE
C        G        SET IN CALLING SUBROUTINE FURPROP TO DESN(KAN) * VAR(KAN)
C        SQB      ON EXIT SQUARED BIAS FOR CANDIDATE KAN
C        (OTHER PARAMETERS IN LIST AND COMMON DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION FREG(MAXNREG, K), DESN(NREG), COVPAR(MAXK, K),
     &   F(MAXK)
C
      COMMON /AM/   AVESQB, SCALEB, FQQUCF(MAXNREG), FQQF(MAXNREG),
     &              QQUC(MAXK, MAXK), QQ(MAXK, MAXK)
C
C        IF FIRST CANDIDATE COMPUTE QQ
C
      IF (KAN .NE. 1) GO TO 50
      DO 27 I = 1, K
      DO 27 J = 1, I
   27 QQ(I, J) = 0.0
      DO 28 L = 1, NREG
      IF (DESN(L) .LT. EPS) GO TO 28
      DO 29 J = 1, K
   29 F(J) = FREG(L, J) * DESN(L)
      DO 30 I = 1, K
      DO 30 J = 1, I
   30 QQ(I, J) = QQ(I, J) + F(I) * F(J)
   28 CONTINUE
      DO 32 I = 1, K
      DO 32 J = 1, I
   32 QQ(J, I) = QQ(I, J)
      CALL PREPOST (QQ, COVPAR, K)
C
C        COMPUTE SQB
C
   50 DO 1 J = 1, K
    1 F(J) = FREG(KAN, J)
      CALL QUAD (F, QQ, MAXK, K, SQB)
      SQB = SQB + 1.0 - 2.0 * G
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE WSMAT (S, IDIM, K, NONLINE)
C
C        WRITE K X K SYMMETRIC MATRIX S USING NONLINE NUMBERS ACROSS PAGE
C
      DIMENSION S(IDIM, K)
C
      DO 3 M = 1, (K + NONLINE - 1) / NONLINE
      NLOW = 1 + (M - 1) * NONLINE
      NUP = NONLINE * M
      DO 4 I = NLOW, K
      NUPMIN = NUP
      IF (I .LT. NUPMIN) NUPMIN = I
    4 WRITE (2, 100) I, NONLINE, ( S(I, J), J = NLOW, NUPMIN)
  100 FORMAT (I4, =E12.4)
      WRITE (2, 102) NONLINE, (J, J = NLOW, NUPMIN)
  102 FORMAT (4X, =I12)
      IF (M .NE. (K + NONLINE - 1) / NONLINE) WRITE (2, 103)
  103 FORMAT (1H )
    3 CONTINUE
C
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE BANDB (CONS, XREG, NREG, NV, FREG, K, DESN, COVPAR,
     &   VAR, DUM1, DUM2, IOUT, NDES, N, ALPHA, GAMMA, NODBEST, IFAIL)
C
C        FIND A MAXIMUM OF MAXNDES DESIGNS WITH DETPAR LESS THAN
C        (1 + GAMMA) TIMES THE D-OPTIMAL DETPAR.  THE NDES DESIGNS
C        ACTUALLY FOUND WILL BE ON MASS STORAGE DEVICE TAPE7 AND
C        THE NODE NUMBERS AT WHICH THEY WERE FOUND WILL BE IN NODBEST
C
```

```
C       DUM1    NOT USED: THIS SPACE NOT NECESSARY FOR D-OPTIMALITY
C               CRITERION BUT DEMANDED BY SOME MORE GENERAL CALLED
C               SUBROUTINES
C       DUM2    AS DUM1
C       IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C               231   ALL CONSTRAINTS DEFINE EQUALITY
C               (ANY OTHER NONZERO VALUE MEANS FAILURE IN A SUBROUTINE
C               CALLED FROM BANDB)
C       (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM, BUT NOTE
C       ON ENTRY AND EXIT DESN, COVPAR, AND VAR ARE NOT DEFINED; THEY
C       ARE PASSED ONLY AS WORKSPACE)
C
      DIMENSION CONS(2, NREG), XREG(MAXNREG, NV), FREG(MAXNREG, K),
     &   DESN(NREG), COVPAR(MAXK, K), VAR(NREG), DUM1(MAXK, K),
     &   DUM2(NREG), NODBEST(MAXNDES), COVPLC(MAXK, MAXK),
     &   VARLC(MAXNREG), XTX(MKP1, MAXK), F(MAXK),
     &   IPERM(MAXNSYM, MAXNREG), IEXTRA(MAXNREG)
C
C       INITIALIZE COVPLC, VARLC, DETLC - THE ANALOGUES OF
C       COVPAR, VAR AND DETPAR WHERE DESIGN IS REPLACED BY THE
C       LOWER CONSTRAINTS CONS(1, .).  THESE WILL BE USED IN
C       COMPUTING BOUND1.
C
      DO 2 L = 1, NREG
    2 DESN(L) = CONS(1, L)
      CALL INITARR (FREG, NREG, K, DESN, COVPLC, DETLC,
     &   VARLC, DUM1, DUM2, 0.0, ALPHA, N, 2HDO, 0, 0, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C       INITIALIZE DESN TO AN APPROXIMATE DESIGN SATISFYING
C       THE CONSTRAINTS AND INITIALIZE COVPAR, VAR AND DETPAR
C
      CALL INITDES (CONS, NREG, K, N, DESN, 0, 0, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      CALL INITARR (FREG, NREG, K, DESN, COVPAR, DETPAR,
     &   VAR, DUM1, DUM2, 0.0, ALPHA, N, 2HDO, 0, 0, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C       INITIALIZE FOLLOWING
C       --------------------
C       NODE    IDENTIFYING NUMBER FOR NODE
C       NANC    ANCESTOR OF CURRENT NODE
C       BEST    SMALLEST CURRENTLY KNOWN DETPAR FOR EXACT DESIGN
C       NDES    NUMBER OF STORED NEARLY-BEST DESIGNS
C       NWAIT   NUMBER OF NODES STACKED ON TAPE8 AWAITING ATTENTION
C       NASS    NUMBER OF OBSERVATIONS ASSIGNED BY LOWER CONSTRAINTS
C       NEXCL   NUMBER OF OBSERVATIONS EXCLUDED BY UPPER CONSTRAINTS
C       IGET    IGET = 1 GET NEXT NODE FROM STACK
C               IGET = 0 NEXT NODE ALREADY IN WORKING ARRAYS
C       IHEAD   IHEAD = 1 WRITE HEADINGS WHEN NEXT NODE OUTPUT
C               IHEAD = 0 DO NOT WRITE HEADINGS
C       KANCH   CANDIDATE AT WHICH A CONSTRAINT IS CHANGED IN
C               DERIVING CURRENT NODE FROM ANCESTOR NODE
C
      NODE = 0
      NANC = 0
      BEST = BIG
      NDES = 0
      NWAIT = 0
      NASS = 0
      NEXCL = 0
      DO 9 L = 1, NREG
      NASS = NASS + IFIX ( (CONS(1, L) + EPS) * FLOAT(N) )
    9 NEXCL = NEXCL + IFIX ( (1.0 - CONS(2, L) + EPS) * FLOAT (N) )
      IGET = 0
      IHEAD = 1
      KANCH = 0
C
C       FIND SYMMETRIES IN DESIGN REGION / MODEL
C       NSYM    NUMBER OF SYMMETRIES
C       IPERM   PERMUTATIONS OF CANDIDATES ASSOCIATED WITH SYMMETRIES
C
      CALL SYMS (XREG, NREG, NV, FREG, K, NSYM, IPERM, IFAIL)
      IF (IFAIL . NE. 0) RETURN
C
C       NODE COMPUTATIONS BEGIN
C       -----------------------
C
C       IF IGET = 1 GET NEW NODE FROM STACK - IF NO NODES RETURN
C
```

```
   10 IF (IGET .EQ. 1 .AND. NWAIT .EQ. 0) RETURN
      IF (IGET .EQ. 1 .AND. NWAIT .GE. 1) CALL STACK (-1, NWAIT,
     &   CONS, NREG, DESN, COVPAR, K, VAR, DETPAR, COVPLC, VARLC,
     &   DETLC, NANC, NASS, NEXCL, KANCH, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
      NODE = NODE + 1
C
C        NUPD IS NUMBER OF TIMES WORKING ARRAYS UPDATED FOR CURRENT NODE
C
      NUPD = 0
C
C        IF IGET = 0 AND NOT FIRST NODE LOWER CONSTRAINT INCREASED AT KANCH
C        TO OBTAIN CURRENT NODE FROM ANCESTOR SO UPDATE COVPLC, VARLC AND DETLC
C
      IF (NODE .EQ. 1 .OR. IGET .NE. 0) GO TO 16
      CONS(1, KANCH) = CONS(1, KANCH) + 1.0 / FLOAT(N)
      NASS = NASS + 1
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPLC, DETLC, VARLC,
     &   DUM1, DUM2, 0.0, KANCH, 1.0 / FLOAT(N), 2HDO, 0, 1,
     &   NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        IF IGET = 1 UPPER CONSTRAINT DECREASED AT KANCH.  USING SYMMETRIES
C        FIND EXTRA CANDIDATES AT WHICH UPPER CONSTRAINT MAY BE DECREASED.
C        NEXTRA  NUMBER OF EXTRA CANDIDATES
C        IEXTRA  IEXTRA(L) IS LTH EXTRA CANDIDATE (L = 1, NEXTRA)
C
   16 IF (IGET .NE. 1) GO TO 19
      NEXTRA = 0
      IF (NSYM .GT. 0) CALL EXTCON (NSYM, IPERM, CONS, NREG, NEXTRA,
     &   IEXTRA, KANCH)
      NEXCL = NEXCL + (NEXTRA + 1) * IFIX ((CONS(2, KANCH)
     &   - CONS(1, KANCH) + EPS) * FLOAT(N) )
      CONS(2, KANCH) = CONS(1, KANCH)
      IF (NEXTRA .EQ. 0) GO TO 60
      DO 61 I = 1, NEXTRA
      KAN = IEXTRA(I)
   61 CONS(2, KAN) = CONS(1, KAN)
   60 IF (N * NREG - NEXCL .GE. N) GO TO 19
C
C        CONSTRAINTS DO NOT ALLOW N-POINT DESIGN: WRITE NODE INFORMATION AND
C        AND GET NEXT NODE
C
      IF (IOUT .EQ. 1HY) CALL WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, 0.0, 1H , 0.0, 1H , 0, 0)
      IGET = 1
      GO TO 10
C
C        COMPUTE BOUND1, FIRST LOWER BOUND ON DETPAR OF EXACT N-POINT DESIGN
C        SATISFYING CONSTRAINTS.  MAXVLC IS CANDIDATE AT WHICH
C        VARLC IS MAXIMIZED.
C
   19 MAXVLC = 0
      VMX = 0.0
      DO 21 L = 1, NREG
      IF (CONS(2, L) - CONS(1, L) .LT. EPS
     &   .OR. VARLC(L) .LT. VMX) GO TO 21
      MAXVLC = L
      VMX = VARLC(L)
   21 CONTINUE
      IF (MAXVLC .NE. 0) GO TO 23
      IFAIL = 231
      RETURN
   23 BOUND1 = DETLC / (1.0 + VMX / FLOAT(N)) ** (N - NASS)
C
C        IF BOUND1 > BEST * (1 + GAMMA) NO BRANCHING: WRITE NODE INFORMATION
C        AND GET NEXT NODE
C
      IF (BOUND1 .LT. BEST * (1.0 + GAMMA) * (1.0 + EPS)) GO TO 25
      IF (IOUT .EQ. 1HY) CALL WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, BOUND1, 1H*, 0.0, 1H , NUPD, 0)
      IGET = 1
      GO TO 10
C
C        IF NASS = N - 1 BOUND1 IS EQUALITY (IGNORING ALPHA TERMS)
C        SO WRITE NODE, UPDATE LIST OF BEST DESIGNS AND GET NEW NODE
C
```

```
   25 IF (NASS .LT. N - 1) GO TO 30
      IF (IOUT .EQ. 1HY) CALL WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, BOUND1, 1H , 0.0, 1H , NUPD, 0)
      CALL BLIST (IOUT, IHEAD, NDES, BEST, GAMMA,
     &   NODE, NODBEST, DETLC, VARLC, N, DESN, CONS, NREG,
     &   NSYM, IPERM)
      IGET = 1
      GO TO 10
C
C        BOUND1 HAS FAILED SO COMPUTE BOUND2: UPDATE DESN FOR CHANGE IN
C        CONSTRAINTS AND FIND OPTIMAL APPROXIMATE DESIGN
C
   30 IF (KANCH .NE. 0) CALL UPCON (CONS, NREG, FREG, K, DESN,
     &   COVPAR, DETPAR, VAR, DUM1, DUM2, KANCH, IEXTRA, NEXTRA, IGET,
     &   NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      CALL APPROX (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   DUM1, DUM2, 0.0- 2HDO, BOUND2, NUPD,
     &   BEST * (1.0 + GAMMA), 2HBR, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        IF BOUND2 > BEST * (1 + GAMMA) NO BRANCHING: WRITE NODE INFORMATION
C        AND GET NEW NODE
C
      IF (BOUND2 .LT. BEST * (1.0 + GAMMA) * (1.0 + EPS)) GO TO 32
      IF (IOUT .EQ. 1HY) CALL WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, BOUND1, 1H , BOUND2, 1H*, NUPD, 0)
      IGET = 1
      GO TO 10
C
C        BOTH BOUNDS HAVE FAILED SO BRANCH ON CANDIDATE MAXVLC:
C        WRITE NODE INFORMATION; PLACE NODE INFORMATION ON STACK,
C        TO BE PROCESSED LATER WITH DECREASED UPPER CONSTRAINT
C        AT MAXVLC; NEXT NODE PROCESSED IS CURRENT NODE BUT WITH
C        INCREASED LOWER CONSTRAINT AT MAXVLC.
C
   32 IF (IOUT .EQ. 1HY) CALL WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, BOUND1, 1H , BOUND2, 1H , NUPD, MAXVLC)
      CALL STACK (1, NWAIT, CONS, NREG, DESN, COVPAR, K,
     &   VAR, DETPAR, COVPLC, VARLC, DETLC, NODE,
     &   NASS, NEXCL, MAXVLC, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      IGET = 0
      KANCH = MAXVLC
      NANC = NODE
      GO TO 10
C
      END
C
C     ---------------------
C
      SUBROUTINE WRNODE (IHEAD, NODE, NANC, NWAIT,
     &   NASS, NEXCL, BOUND1, ISUC1, BOUND2, ISUC2, NUPD, MAXVLC)
C
C        WRITE NODE INFORMATION
C
C        IHEAD   1  HEADINGS NEEDED
C                0  NO HEADINGS
C        BOUND1  FIRST LOWER BOUND ON DETPAR OF EXACT N-POINT DESIGN SATISYING
C                CONSTRAINTS
C        ISUC1   1H   BOUND1 FAILS
C                1H*  BOUND1 SUCCEEDS IN PREVENTING BRANCHING
C        BOUND2  SECOND LOWER BOUND
C        ISUC2   1H   BOUND2 FAILS
C                1H*  BOUND2 SUCCEEDS IN PREVENTING BRANCHING
C        NODE, NANC, NWAIT, NASS, NEXCL, NUPD, AND MAXVLC DESCRIBED
C        IN CALLING SUBROUTINE BANDD
C
C        IF IHEAD = 1 WRITE HEADINGS FOR NODE INFORMATION
C
      IF (IHEAD .EQ. 1) WRITE (2, 100)
  100 FORMAT (// 27X, 'NO. OF OBS.'  / 27X, 'ASSIGNED AND' /
     &   7X, 'ANCEST.', 3X, 'STACK', 5X, 'EXCLUDED BY', 5X, 'BOUND1',
     &   6X, 'BOUND2', 3X, 'UPDATE', 2X, 'BRANCH' /
     &   2X, 'NODE', 4X, 'NODE', 4X, 'SIZE', 5X, 'CONSTRAINTS', 2X,
     &   '*=SUCCESS', 3X, '*=SUCCESS', 4X, 'CALLS', 6X, 'ON' /)
```

```
C
      IHEAD = 0
C
C        WRITE NODE INFORMATION
C
      WRITE (2, 101) NODE, NANC, NWAIT, NASS, NEXCL,
     &   BOUND1, ISUC1, BOUND2, ISUC2, NUPD, MAXVLC
  101 FORMAT (I6, 4I8, 2(E11.4, A1), 2I8)
C
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE BLIST (IOUT, IHEAD, NDES, BEST, GAMMA,
     &   NODE, NODBEST, DETLC, VARLC, N, DESN, CONS, NREG,
     &   NSYM, IPERM)
C
C        BLIST MANAGES A LIST OF A MAXIMUM OF MAXNDES BEST DESIGNS
C
C        IOUT, GAMMA, NODBEST, N, DESN, CONS AND NREG DESCRIBED IN
C        MAIN PROGRAM
C        IHEAD, NDES, BEST, NODE, DETLC AND VARLC DESCRIBED IN BANDB
C
      DIMENSION NODBEST(MAXNDES), VARLC(MAXNREG), DESN(NREG),
     &   CONS(2, NREG), DETPAR(MAXNDES), WORKSP(MAXNREG),
     &   IPERM(MAXNSYM, NREG), IUSED(MAXNSYM)
C
C        INEW IS FLAG INDICATING WHETHER BEST (LOWEST DETPAR OF
C        AN N-POINT DESIGN) HAS BEEN REPLACED
C
       INEW = 0
C
C        SET DESN TO N - 1 POINT DESIGN DEFINED BY CONS(1, .)
C
      DO 2 L = 1, NREG
    2 DESN(L) = CONS(1, L)
C
      DO 3 L = 1, NREG
      IF (CONS(2, L) - CONS(1, L) .LT. EPS) GO TO 3
C
C        DETPARL IS DETPAR FOR DESIGN WHEN CANDIDATE L IS ADDED
C        TO THE N - 1 POINTS
C
      DETPARL = DETLC / (1.0 + VARLC(L) / FLOAT(N))
C
C        REJECT DESIGN IF DETPARL > BEST * (1 + GAMMA)
C
      IF (DETPARL .GT. BEST * (1.0 + GAMMA) * (1.0 + EPS)) GO TO 3
C
C        ADD NEW POINT AT CANDIDATE L TO DESIGN
C
      DESN(L) = DESN(L) + 1.0 / FLOAT(N)
C
      IF (NDES .GT. 0) GO TO 10
      M = 1
      GO TO 9
C
C        FIND FIRST M SUCH THAT DETPAR(M) > DETPARL
C
   10 DO 7 M = 1, NDES
      IF (DETPAR(M) .LT. DETPARL * (1.0 - EPS)) GO TO 7
      IF (DETPAR(M) .GT. DETPARL * (1.0 + EPS)) GO TO 6
C
C        DETPAR(M) = DETPARL SO CHECK NOT SAME DESIGN
C
      CALL MS7MAN (M, WORKSP, NREG, -1)
      DO 5 LL = 1, NREG
      IF (ABS(WORKSP(LL) - DESN(LL)) .GT. EPS) GO TO 30
    5 CONTINUE
C
C        SAME DESIGN SO REJECT
C
      GO TO 18
C
C        ALSO CHECK NOT DIFFERING ONLY BY SYMMETRY
C
C        THERE ARE 2 ** NSYM - 1 COMBINATIONS OF THE NSYM BASIC SYMMETRIES
C        WHERE EACH IS OR IS NOT USED AND AT LEAST ONE IS USED.  ALL SUCH
C        COMBINATIONS DEFINE SYMMETRIES.
```

```
C        IUSED(MM)   1 SYMMETRY MM USED (M = 1, NSYM)
C                    0 SYMMETRY MM NOT USED
C
   30 IF (NSYM .EQ. 0) GO TO 7
      DO 31 MM = 1, NSYM
   31 IUSED(MM) = 0
C
      DO 33 JJJ = 1, 2 ** NSYM - 1
C
      IUSED(1) = IUSED(1) + 1
      IF (NSYM .EQ. 1) GO TO 39
      DO 38 MM = 1, NSYM - 1
      IF (IUSED(MM) .NE. 2) GO TO 39
      IUSED(MM) = 0
      IUSED(MM + 1) = IUSED(MM + 1) + 1
   38 CONTINUE
C
   39 DO 40 LL = 1, NREG
      LPERM = LL
      DO 41 MM = 1, NSYM
      IF (IUSED(MM) .EQ. 1) LPERM = IPERM(MM, LPERM)
   41 CONTINUE
      IF (ABS (WORKSP(LL) - DESN(LPERM)) .GT. EPS) GO TO 33
   40 CONTINUE
C
C        DIFFERS ONLY BY SYMMETRY SO REJECT
C
      GO TO 18
C
   33 CONTINUE
C
    7 CONTINUE
C
C        NEW DESIGN INFERIOR TO ALL ON LIST - IF
C        LIST FULL REJECT, IF NOT FULL ADD AT END
C
      IF (NDES .EQ. MAXNDES) GO TO 18
      M = NDES + 1
      GO TO 9
C
C        IF LIST FULL REMOVE LAST DESIGN
C
    6 IF (NDES .EQ. MAXNDES) NDES = NDES - 1
C
C        MOVE DESIGNS M, ..., NDES DOWN ONE POSITION
C
      IF (M .GT. NDES) GO TO 9
      DO 8 II = M, NDES
      JJ = NDES + M - II
      DETPAR(JJ + 1) = DETPAR(JJ)
      NODBEST(JJ + 1) = NODBEST(JJ)
      CALL MS7MAN (JJ, WORKSP, NREG, -1)
    8 CALL MS7MAN (JJ + 1, WORKSP, NREG, 1)
C
C        INSERT NEW DESIGN AT POSITION M
C
    9 DETPAR(M) = DETPARL
      NODBEST(M) = NODE
      CALL MS7MAN (M, DESN, NREG, 1)
      NDES = NDES + 1
C
C        REMOVE THE POINT AT CANDIDATE L FROM DESN
C
   18 DESN(L) = DESN(L) - 1.0 / FLOAT(N)
C
C        UPDATE BEST IF NECESSARY
C
      IF (DETPARL .GT. BEST * (1.0 - EPS)) GO TO 3
      BEST = DETPARL
      INEW = 1
C
    3 CONTINUE
C
C        IF BEST HAS NOT BEEN REPLACED RETURN
C
      IF (INEW .EQ. 0) RETURN
C
C        IF IOUT = 1HY WRITE NEW.BEST
C
      IF (IOUT .EQ. 1HY) WRITE (2, 100) BEST
  100 FORMAT (/ ' OPTIMAL DETPAR REPLACED BY', E12.4)
      IHEAD = 1
```

```
C
C        BEST HAS BEEN DECREASED SO SOME DESIGNS ON LIST MAY NOT HAVE
C        DETPAR < BEST * (1 + GAMMA): REDEFINE NDES
C
      DO 50 M = 1, NDES
      IF (DETPAR(M) .LT. BEST * (1.0 + GAMMA) * (1.0 + EPS))
     &   GO TO 50
      NDES = M - 1
      RETURN
   50 CONTINUE
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE STACK (IWR, NWAIT, CONS, NREG, DESN, COVPAR, K,
     &   VAR, DETPAR, COVPLC, VARLC, DETLC, NODE,
     &   NASS, NEXCL, MAXVLC, IFAIL)
C
C        STACK MANAGES A STACK OF NODES ON SEQUENTIAL FILE TAPE8
C        IF IWR = 1 WRITE NODE INFORMATION IN RECORD NWAIT + 1
C        AND INCREMENT NWAIT
C        IF IWR = -1 READ NODE INFORMATION FROM RECORD NWAIT
C        AND DECREMENT NWAIT
C
C        NODE     IF IWR = -1 NODE WILL BE ANCESTOR NODE IDENTIFIER,
C                 IF IWR = 1 NODE IS CURRENT NODE IDENTIFIER
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                 241   ATTEMPT TO READ NON-EXISTENT NODE
C                 242   NEITHER READ NOR WRITE INDICATED
C        CONS, NREG, DESN, COVPAR, K, VAR AND DETPAR DESCRIBED IN MAIN PROGRAM
C        COVPLC, VARLC, DETLC, NASS, NEXCL AND MAXVLC DESCRIBED IN CALLING
C        SUBROUTINE BANDB
C
      DIMENSION CONS(2, NREG), DESN(NREG), COVPAR(MAXK, K),
     &   VAR(NREG), COVPLC(MAXK, K), VARLC(NREG)
C
C        IF NWAIT = 0 REWIND TAPE 8 AND CHECK NOT READING
C
      IF (NWAIT .GT. 0) GO TO 3
      REWIND 8
      IF (IWR .NE. -1) GO TO 20
      IFAIL = 241
      RETURN
C
C        IF IWR = -1 BACKSPACE AND READ RECORD NWAIT
C
    3 IF (IWR .NE. -1) GO TO 20
      BACKSPACE 8
      READ (8) ((CONS(J, L), J=1, 2), L = 1, NREG),
     &   (DESN(L), L = 1, NREG), ((COVPAR(I, J), J = 1, I), I = 1, K),
     &   (VAR(L), L = 1, NREG), DETPAR,
     &   ((COVPLC(I, J), J = 1, I), I = 1, K), (VARLC(L), L = 1, NREG),
     &   DETLC, NODE, NASS, NEXCL, MAXVLC
      IF (EOF(8) .EQ. 0.0) GO TO 15
      IFAIL = 241
      RETURN
   15 DO 16 I = 1, K
      DO 16 J = 1, I
      COVPAR(J, I) = COVPAR(I, J)
   16 COVPLC(J, I) = COVPLC(I, J)
C
C        BACKSPACE AGAIN AS NOW NWAIT - 1 RECORDS ON STACK
C
      BACKSPACE 8
      NWAIT = NWAIT - 1
      RETURN
C
C        IF IWR = 1 WRITE NODE INFORMATION AT RECORD NWAIT + 1
C
   20 IF (IWR .NE. 1) GO TO 30
      WRITE (8) ((CONS(J, L), J = 1, 2), L = 1, NREG),
     &   (DESN(L), L = 1, NREG), ((COVPAR(I, J), J = 1, I), I = 1, K),
     &   (VAR(L), L = 1, NREG), DETPAR,
     &   ((COVPLC(I, J), J = 1, I), I = 1, K), (VARLC(L), L = 1,NREG),
     &   DETLC, NODE, NASS, NEXCL, MAXVLC
      NWAIT = NWAIT + 1
      RETURN
```

```
C
C        IWR NOT EQUAL TO -1 OR 1
C
   30 IFAIL = 242
      RETURN
      END
C
C     --------------------
C
      SUBROUTINE UPCON (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR,
     &   VAR, DUM1, DUM2, KAN, IEXTRA, NEXTRA, IGET, NUPD, IFAIL)
C
C        CONSTRAINTS HAVE BEEN CHANGED AT CANDIDATE KAN AND CANDIDATES
C        IEXTRA(I) (I = 1, NEXTRA).  UPDATE DESN TO SATISFY CONSTRAINTS.
C
C        IFAIL   ON ENTRY 0.  ON EXIT 0 O.K., FAILURE INDICATED BY
C                251   NO DESIGN CAN SATISFY CONSTRAINTS
C                252   IGET NOT EQUAL TO 0 OR 1
C                (ANY OTHER NONZERO VALUE MEANS FAILURE IN A SUBROUTINE
C                CALLED FROM UPCON)
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
C
      DIMENSION CONS(2, NREG), FREG(MAXNREG, K), DESN(NREG),
     &   COVPAR(MAXK, K), VAR(NREG), DUM1(MAXK, K), DUM2(NREG),
     &   IEXTRA(MAXNREG)
C
      IF (IGET .EQ. 0) GO TO 200
      IF (IGET .EQ. 1) GO TO 300
      IFAIL = 252
      RETURN
C
C        LOWER CONSTRAINT POSSIBLY VIOLATED
C        ----------------------------------
C
  200 IF (DESN(KAN) .GT. CONS(1, KAN) - EPS) RETURN
C
C        INCREASE DESN(KAN) TO LOWER CONSTRAINT BY ADDING VIOL
C
      VIOL = CONS(1, KAN) - DESN(KAN)
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   DUM1, DUM2, 0.0, KAN, VIOL, 2HDO, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
C
C        A TOTAL OF VIOL MUST BE REMOVED FROM DESN OF OTHER CANDIDATES.
C        FIND CANDIDATE MIN ACHIEVING MINIMUM VAR SUBJECT TO SLACK AT
C        LOWER CONSTRAINT AND REMOVE VIOL FROM DESN(MIN) IF SLACK ALLOWS.
C        IF HOWEVER SLACK < VIOL REMOVE SLACK AND FIND A FURTHER MIN ETC.
C
  210 MIN = 0
      AMIN = BIG
      DO 211 L = 1, NREG
      IF (DESN(L) - CONS(1, L) .LT. EPS
     &   .OR. VAR(L) .GT. AMIN) GO TO 211
      MIN = L
      AMIN = VAR(L)
  211 CONTINUE
      IF (MIN .NE. 0) GO TO 214
      IFAIL = 251
      RETURN
  214 SLACK = DESN(MIN) - CONS(1, MIN)
      REDUC = VIOL
      IF (VIOL .GT. SLACK) REDUC = SLACK
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   DUM1, DUM2, 0.0, MIN, -REDUC, 2HDO, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      VIOL = VIOL - REDUC
      IF (VIOL .GT. EPS) GO TO 210
      RETURN
C
C        UPPER CONSTRAINT(S) POSSIBLY VIOLATED
C        -------------------------------------
C
  300 VIOL = 0.0
      DO 301 I = 1, NEXTRA + 1
      IF (I .LE. NEXTRA) L = IEXTRA(I)
      IF (I .EQ. NEXTRA + 1) L = KAN
      IF (DESN(L) .LT. CONS(2, L) + EPS) GO TO 301
```

```
C
C        DECREASE DESN(L) TO UPPER CONSTRAINT BY REMOVING VIOLI
C
      VIOLI = DESN(L) - CONS(2, L)
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   DUM1, DUM2, 0.0, L, -VIOLI, 2HDO, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      VIOL = VIOL + VIOLI
  301 CONTINUE
      IF (VIOL .LT. EPS) RETURN
C
C        A TOTAL OF VIOL MUST BE ADDED TO DESN OF OTHER CANDIDATES.
C        FIND CANDIDATE MAX ACHIEVING MAXIMUM VARIANCE SUBJECT TO SLACK
C        AT UPPER CONSTRAINT AND ADD VIOL TO DESN(MAX) IF SLACK ALLOWS.
C        IF HOWEVER SLACK < VIOL ADD SLACK AND FIND A FURTHER MAX ETC.
C
  310 MAX = 0
      AMAX = 0.0
      DO 311 L = 1, NREG
      IF (CONS(2, L) - DESN(L) .LT. EPS
     &   .OR. VAR(L) .LT. AMAX) GO TO 311
      MAX = L
      AMAX = VAR(L)
  311 CONTINUE
      IF (MAX .NE. 0) GO TO 314
      IFAIL =251
      RETURN
  314 SLACK = CONS(2, MAX) - DESN(MAX)
      ADD = VIOL
      IF (VIOL .GT. SLACK) ADD = SLACK
      CALL UPDATE (CONS, NREG, FREG, K, DESN, COVPAR, DETPAR, VAR,
     &   DUM1, DUM2, 0.0, MAX, ADD, 2HDO, 0, 0, NUPD, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      VIOL = VIOL - ADD
      IF (VIOL .GT. EPS) GO TO 310
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE MORE (IREPLY, INTER)
C
C        READ IREPLY
C
C        IREPLY AND INTER DESCRIBED IN MAIN PROGRAM
C
      DATA MAXREAD /4/
      IF (INTER .NE. 1) MAXREAD = 1
C
      DO 4 NN = 1, MAXREAD
      IF (INTER .EQ. 1) WRITE (2, 100)
  100 FORMAT (/ " START AGAIN WITH SAME DESIGN REGION (YES OR NO)" /)
      READ (1, 101) IREPLY
  101 FORMAT (A1)
      IF (EOF(1) .EQ. 0.0) GO TO 5
      WRITE (2, 120)
  120 FORMAT (/ " ERROR: END OF FILE WHEN READING START AGAIN" )
      GO TO 4
    5 IF (IREPLY .EQ. 1HY .OR. IREPLY .EQ. 1HN) RETURN
      WRITE (2, 105)
  105 FORMAT (/ " ERROR: Y OR N EXPECTED TO ANSWER START AGAIN" )
    4 CONTINUE
      STOP
      END
C
C     ---------------------
C
      SUBROUTINE SYMS (XREG, NREG, NV, FREG, K, NSYM, IPERM, IFAIL)
C
C        FIND SYMMETRIES IN DESIGN REGION / MODEL
C
C        NSYM     ON EXIT NUMBER OF SYMMETRIES FOUND
C        IPERM    ON EXIT IPERM(M, L) IS CANDIDATE IN LTH PERMUTED POSITION
C                 FOR SYMMETRY M (M = 1, NSYM; L = 1, NREG)
C        IFAIL    ON ENTRY 0.  ON EXIT 0 O.K., NONZERO VALUE MEANS FAILURE
C                 IN A SUBROUTINE CALLED FROM SYMS
C        (OTHER PARAMETERS IN LIST DESCRIBED IN MAIN PROGRAM)
C
```

```
      DIMENSION XREG(MAXNREG, NV), FREG(MAXNREG, K),
     &   IPERM(MAXNSYM, NREG), ISYM(MAXNV, MAXNV), X(MAXNV),
     &   UINV(MAXK, MAXK), V(MAXK, MAXK), A(MAXK, MAXK)
C
C        COMMON /SPACE1/ USED ONLY TO REDUCE STORAGE, NOT FOR COMMUNICATION
C
      COMMON /SPACE1/ U(MKP1, MAXK)
C
C        WE SHALL NEED UINV, THE INVERSE OF U WHERE U = TRANSPOSE OF FREG * FREG
C
      DO 7 I = 1, K
      DO 7 J = I, K
      U(I, J) = 0.0
      DO 7 L = 1, NREG
    7 U(I, J) = U(I, J) + FREG(L, I) * FREG(L, J)
      CALL INFINV (U, UINV, K, IFAIL)
      IF (IFAIL .NE. 0) RETURN
      DO 8 I = 1, K
      DO 8 J = 1, I
    8 UINV(J, I) = UINV(I, J)
C
C        THE POSSIBLE SYMMETRIES INVESTIGATED ARE INDEXED (II, JJ):
C        (II, II) IS REFLECTION OF EXPLANATORY VARIABLE II ABOUT ZERO
C        (II, JJ) IS INTERCHANGE OF EXPLANATORY VARIABLES II AND JJ (JJ < II)
C        SUCH SYMMETRIES, IF THEY EXIST, ALWAYS PERMUTE THE CANDIDATES
C
      NSYM = 0
      DO 10 II = 1, NV
      DO 10 JJ = 1, II
C
C        SET ISYM(II, JJ) = 0.  IF SYMMETRY EXISTS IT WILL BE SET TO 1.
C
      ISYM(II, JJ) = 0
C
C        V WILL BE TRANSPOSE OF PERMUTED ROWS OF FREG * FREG
C
      DO 11 I = 1, K
      DO 11 J = 1, K
   11 V(I, J) = 0.0
C
      DO 12 L = 1, NREG
C
C        APPLY REFLECTION / INTERCHANGE TO CANDIDATE L
C
      DO 13 I = 1, NV
   13 X(I) = XREG(L, I)
      IF (II .NE. JJ) GO TO 14
      X(II) = -X(II)
      GO TO 15
   14 XS = X(II)
      X(II) = X(JJ)
      X(JJ) = XS
C
C        SEARCH FOR REFLECTED / INTERCHANGED X IN XREG
C
   15 DO 18 LL = 1, NREG
      DO 19 I = 1, NV
      IF (ABS (XREG(LL, I) - X(I)) .GT. EPS) GO TO 18
   19 CONTINUE
C
C        CURRENT XREG(LL, .) = X(.) SO SAVE LL IN IPERM(NSYM + 1, L),
C        INCREMENT V AND GO TO NEXT CANDIDATE
C
      IPERM(NSYM + 1, L) = LL
      DO 17 I = 1, K
      DO 17 J = 1, K
   17 V(I, J) = V(I, J) + FREG(LL, I) * FREG(L, J)
      GO TO 12
C
   18 CONTINUE
C
C        FOR NO LL IS XREG(LL, .) = X(.).  NO SYMMETRY.
C
      GO TO 10
C
   12 CONTINUE
```

```
C
C       CHECK FREG(L, .) = A * FREG(LPERM, .) WHERE A IS A K X K MATRIX
C       AND LPERM = IPERM(NSYM + 1, L) (L = 1, NREG).  TRY A = UINV * V.
C       (IF THIS CONDITION HOLDS, THE FURTHER REQUIREMENT THAT SQUARED
C       DETERMINANT OF A = 1 IN BRANCH AND BOUND PAPER IS AUTOMATICALLY
C       SATISFIED).
C
      DO 22 I = 1, K
      DO 22 J = 1, K
      A(I, J) = 0.0
      DO 22 M = 1, K
   22 A(I, J) = A(I, J) + UINV(I, M) * V(M, J)
      DO 24 L = 1, NREG
      LPERM = IPERM(NSYM + 1, L)
      DO 24 I = 1, K
      AFPERM = 0.0
      DO 25 J = 1, K
   25 AFPERM = AFPERM + A(I, J) * FREG(LPERM, J)
      IF (ABS (FREG(L, I) - AFPERM) .GT. EPS) GO TO 10
   24 CONTINUE
C
C       SYMMETRY IS O.K.
C
      NSYM = NSYM + 1
      ISYM(II, JJ) = 1
C
      IF (NSYM .EQ. MAXNSYM) GO TO 300
C
   10 CONTINUE
C
C       WRITE INFORMATION ON SYMMETRIES FOUND
C
  300 CALL WRSYM (ISYM, NV, NSYM)
      RETURN
      END
C
C     ---------------------
C
      SUBROUTINE WRSYM (ISYM, NV, NSYM)
C
C       OUTPUT INFORMATION ON SYMMETRIES DETECTED
C
      DIMENSION ISYM(MAXNV, NV)
C
      WRITE (2, 100) NSYM
  100 FORMAT (/ ' SYMMETRIES DETECTED', I10)
C
      DO 1 I = 1, NV
      DO 1 J = 1, I
      IF (ISYM(I, J) .EQ. 0) ISYM(I, J) = 1HN
      IF (ISYM(I, J) .EQ. 1) ISYM(I, J) = 1HY
    1 CONTINUE
C
      WRITE (2, 103) NV, (I, I = 1, NV)
  103 FORMAT( / ' REFLECTIONS' // =(3X, 'VAR', I2))
      WRITE (2, 104) NV, (ISYM(I, I), I = 1, NV)
  104 FORMAT (=(7X, A1))
C
      IF (NV .EQ. 1) RETURN
      WRITE (2, 110) NV - 1, (J, J = 1, NV - 1)
  110 FORMAT (/ ' INTERCHANGES' // 8X, =(3X, 'VAR', I2))
      DO 7 I = 2, NV
    7 WRITE (2, 112) I, I - 1, (ISYM(I, J), J = 1, I - 1)
  112 FORMAT ( 3X, 'VAR', I2, =(7X, A1))
      RETURN
      END
```

```
      SUBROUTINE EXTCON (NSYM, IPERM, CONS, NREG, NEXTRA,
     &   IEXTRA, KANCH)
C
C       FIND EXTRA CANDIDATES AT WHICH UPPER CONSTRAINT MAY BE
C       DECREASED BECAUSE OF SYMMETRIES
C
C       NEXTRA  ON EXIT NUMBER OF EXTRA CANDIDATES
C       IEXTRA  ON EXIT IEXTRA(L) IS LTH EXTRA CANDIDATE (L = 1, NEXTRA)
C       NSYM AND IPERM DESCRIBED IN SYMS
C       KANCH DESCRIBED IN BANDB
C       CONS AND NREG DESCRIBED IN MAIN PROGRAM
C
      DIMENSION IPERM(MAXNSYM, NREG), CONS(2, NREG),
     &   IEXTRA(MAXNREG), IUSED(MAXNSYM)
C
      NEXTRA = 0
C
C       THERE ARE 2 ** NSYM - 1 COMBINATIONS OF THE NSYM BASIC SYMMETRIES
C       WHERE EACH IS OR IS NOT USED AND AT LEAST ONE IS USED.  ALL SUCH
C       COMBINATIONS DEFINE SYMMETRIES.
C       IUSED(M)    1 SYMMETRY M USED (M = 1, NSYM)
C                   0 SYMMETRY M NOT USED
C
      DO 6 M = 1, NSYM
    6 IUSED(M) = 0
C
      DO 7 JJJ = 1, 2 ** NSYM - 1
C
      IUSED(1) = IUSED(1) + 1
      IF (NSYM .EQ. 1) GO TO 9
      DO 8 M = 1, NSYM - 1
      IF (IUSED(M) .NE. 2) GO TO 9
      IUSED(M) = 0
      IUSED(M + 1) = IUSED(M + 1) + 1
    8 CONTINUE
C
C       FIND KANPERM - PERMUTED POSITION OF CANDIDATE KANCH AFTER
C       THE COMBINATION OF PERMUTATIONS
C
    9 KANPERM = KANCH
      DO 10 M = 1, NSYM
      IF (IUSED(M) .EQ. 1) KANPERM = IPERM(M, KANPERM)
   10 CONTINUE
C
C       IF KANPERM = KANCH OR KANPERM ALREADY STORED IN IEXTRA GO TO
C       NEXT COMBINATION
C
      IF (KANPERM .EQ. KANCH) GO TO 7
      IF (NEXTRA .EQ. 0) GO TO 15
      DO 12 J = 1, NEXTRA
      IF (KANPERM .EQ. IEXTRA(J)) GO TO 7
   12 CONTINUE
C
C       CHECK CONS(1, L) = CONS(1, LPERM) AND CONS(2, L) = CONS(2, LPERM)
C       WHERE LPERM IS CANDIDATE IN LTH PERMUTED POSITION
C       (L = 1, NREG)
C
   15 DO 16 L = 1, NREG
      LPERM = L
      DO 17 M = 1, NSYM
      IF (IUSED(M) .EQ. 1) LPERM = IPERM(M, LPERM)
   17 CONTINUE
      IF (ABS (CONS(1, L) - CONS(1, LPERM)) .GT. EPS
     &   .OR. ABS (CONS(2, L) - CONS(2, LPERM)) .GT. EPS) GO TO 7
   16 CONTINUE
C
C       KANPERM IS A NEW EXTRA CANDIDATE AT WHICH UPPER CONSTRAINT MAY
C       BE DECREASED
C
      NEXTRA = NEXTRA + 1
      IEXTRA(NEXTRA) = KANPERM
C
    7 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FAIL (IFAIL)
C
C        WRITE FAILURE MESSAGE
C
      IF (IFAIL .LE. 0 .OR. IFAIL .GE. 261) GO TO 30
      II = (IFAIL - 1) / 10 + 1
      GO TO (1, 30, 30, 30, 30, 30, 30, 8, 9, 10, 11, 30, 13, 14,
     &   15, 16, 17, 18, 19, 20, 21, 22, 30, 24, 25, 26), II
C
   30 WRITE (2, 4000)
 4000 FORMAT (/ ' ERROR IN SUBROUTINE UNKNOWN')
      WRITE (2, 5000)
 5000 FORMAT (' IFAIL NOT RECOGNISED')
      RETURN
C
    1 WRITE (2, 101)
  101 FORMAT (/ ' ERROR IN MAIN PROGRAM')
      IF (IFAIL .EQ. 1) WRITE (2, 1001)
 1001 FORMAT (' ALGORITHM IS NOT AP, EX OR BR' )
      IF (IFAIL .GT. 1) WRITE (2, 5000)
      RETURN
C
    8 WRITE (2, 108)
  108 FORMAT (/ ' ERROR IN SUBROUTINE OUTDES')
      IF (IFAIL .EQ. 71) WRITE (2, 1071)
 1071 FORMAT (' INSUFFICIENT PAGE WIDTH TO OUTPUT DESIGN')
      IF (IFAIL .EQ. 72) WRITE (2, 1072)
 1072 FORMAT (' CANDIDATE WITH REPLICATION OF 100 OR MORE')
      IF (IFAIL .GT. 72) WRITE (2, 5000)
      RETURN
C
    9 WRITE (2, 109)
  109 FORMAT (/ ' ERROR IN SUBROUTINE LOWBND')
      IF (IFAIL .EQ. 81) WRITE (2, 1081)
 1081 FORMAT (' CRITERION IS NOT DO, AV OR AM')
      IF (IFAIL .EQ. 82) WRITE (2, 1192)
      IF (IFAIL .GT. 82) WRITE (2, 5000)
      RETURN
C
   10 WRITE (2, 110)
  110 FORMAT (/ ' ERROR IN SUBROUTINE MING')
      IF (IFAIL .EQ. 91) WRITE (2, 1091)
 1091 FORMAT(' NO SLACK AT ANY CANDIDATE' )
      IF (IFAIL .EQ. 92) WRITE (2, 1092)
 1092 FORMAT (' MINIMUM AND MAXIMUM AT SAME CANDIDATE')
      IF (IFAIL .GT. 92) WRITE (2, 5000)
      RETURN
C
   11 WRITE (2, 111)
  111 FORMAT (/ ' ERROR IN SUBROUTINE APPROX')
      IF (IFAIL .EQ. 101) WRITE (2, 1101)
 1101 FORMAT (' ALGORITHM DOES NOT CONVERGE IN MAXIT ITERATIONS')
      IF (IFAIL .EQ. 102) WRITE (2, 1081)
      IF (IFAIL .GT. 102) WRITE (2, 5000)
      RETURN
C
   13 WRITE (2, 113)
  113 FORMAT (/ ' ERROR IN SUBROUTINE CRITV')
      IF (IFAIL .EQ. 121) WRITE (2, 1132)
      IF (IFAIL .GT. 121) WRITE (2, 5000)
      RETURN
C
   14 WRITE (2, 114)
  114 FORMAT (/ ' ERROR IN SUBROUTINE EXCURS')
      IF (IFAIL .EQ. 131) WRITE (2, 1191)
      IF (IFAIL .EQ. 132) WRITE (2, 1132)
 1132 FORMAT (' CRITERION IS NOT DO, AV, MV OR AM' )
      IF (IFAIL .GT. 132) WRITE (2, 5000)
      RETURN
C
   15 WRITE (2, 115)
  115 FORMAT (/ ' ERROR IN SUBROUTINE INFINV')
      IF (IFAIL .EQ. 141) WRITE (2, 1141)
 1141 FORMAT (' INFORMATION MATRIX NOT POSITIVE DEFINITE')
      IF (IFAIL .EQ. 142) WRITE (2, 1142)
 1142 FORMAT (' INFORMATION MATRIX ILL-CONDITIONED')
      IF (IFAIL .GT. 142) WRITE (2, 5000)
      RETURN
```

```
C
   16 WRITE (2, 116)
  116 FORMAT (/ ' ERROR IN SUBROUTINE DET')
      IF (IFAIL .EQ. 151) WRITE (2, 1141)
      IF (IFAIL .EQ. 152) WRITE (2, 1152)
 1152 FORMAT (' OVERFLOW - DETERMINANT TOO LARGE')
      IF (IFAIL .EQ. 153) WRITE (2, 1153)
 1153 FORMAT (' UNDERFLOW - DETERMINANT TOO SMALL')
      IF (IFAIL .GT. 153) WRITE (2, 5000)
      RETURN
C
   17 WRITE (2, 117)
  117 FORMAT (/ ' ERROR IN SUBROUTINE OPTRAN')
      IF (IFAIL .EQ. 161) WRITE (2, 1081)
      IF (IFAIL .EQ. 162) WRITE (2, 1162)
 1162 FORMAT (' OPTIMAL TRANSFER LESS THAN EPS')
      IF (IFAIL .EQ. 163) WRITE (2, 1163)
 1163 FORMAT (' PARAMETER OUT OF RANGE WHEN NAG ROUTINE',
     &    ' E04ABF CALLED')
      IF (IFAIL .EQ. 164) WRITE (2, 1164)
 1164 FORMAT (' NO CONVERGENCE AFTER 30 ITERATIONS IN',
     &    ' NAG ROUTINE E04ABF')
      IF (IFAIL .GT. 164) WRITE (2, 5000)
      RETURN
C
   18 WRITE (2, 118)
  118 FORMAT (/ ' ERROR IN SUBROUTINE UPDATE')
      IF (IFAIL .EQ. 171) WRITE (2, 1171)
 1171 FORMAT (' BETA LESS THAN EPS')
      IF (IFAIL .EQ. 172) WRITE (2, 1172)
 1172 FORMAT (' CHANGE VIOLATES CONSTRAINTS')
      IF (IFAIL .EQ. 173) WRITE (2, 1132)
      IF (IFAIL .GT. 173) WRITE (2, 5000)
      RETURN
C
   19 WRITE (2, 119)
  119 FORMAT (/ ' ERROR IN SUBROUTINE INITARR')
      IF (IFAIL .EQ. 181) WRITE (2, 1132)
      IF (IFAIL .GT. 181) WRITE (2, 5000)
      RETURN
C
   20 WRITE (2, 120)
  120 FORMAT (/ ' ERROR IN SUBROUTINE INITDES')
      IF (IFAIL .EQ. 191) WRITE (2, 1191)
 1191 FORMAT (' DESIGN COMPLETELY SPECIFIED BY CONSTRAINTS' )
      IF (IFAIL .EQ. 192) WRITE (2, 1192)
 1192 FORMAT(' NO DESIGN CAN SATISFY CONSTRAINTS' )
      IF (IFAIL .GT. 192) WRITE (2, 5000)
      RETURN
C
   21 WRITE (2, 121)
  121 FORMAT (/ ' ERROR IN SUBROUTINE ADDSUB')
      IF (IFAIL .EQ. 201) WRITE (2, 1201)
 1201 FORMAT (' CONSTRAINTS DO NOT ALLOW THE ADDITION OR',
     &    ' SUBTRACTION' )
      IF (IFAIL .GT. 201) WRITE (2, 5000)
      RETURN
C
   22 WRITE (2, 122)
  122 FORMAT (/ ' ERROR IN SUBROUTINE ORDER')
      IF (IFAIL .EQ. 211) WRITE (2, 1211)
 1211 FORMAT(/ ' NAG ROUTINE M01ABF FAILS')
      IF (IFAIL .GT. 211) WRITE (2, 5000)
      RETURN
C
   24 WRITE (2, 124)
  124 FORMAT (/ ' ERROR IN SUBROUTINE BANDB')
      IF (IFAIL .EQ. 231) WRITE (2,1231)
 1231 FORMAT (/ ' ALL CONSTRAINTS DEFINE EQUALITY' )
      IF (IFAIL .GT. 231) WRITE (2, 5000)
      RETURN
C
   25 WRITE (2, 125)
  125 FORMAT (/ ' ERROR IN SUBROUTINE STACK')
      IF (IFAIL .EQ. 241) WRITE (2, 1241)
 1241 FORMAT (' ATTEMPT TO READ NONEXISTENT NODE')
      IF (IFAIL .EQ. 242) WRITE (2, 1242)
 1242 FORMAT (' NEITHER READ NOR WRITE INDICATED' )
      IF (IFAIL .GT. 242) WRITE (2, 5000)
      RETURN
C
```

```
  26 WRITE (2, 126)
 126 FORMAT (/ ' ERROR IN SUBROUTINE UPCON')
     IF (IFAIL .EQ. 251) WRITE (2, 1192)
     IF (IFAIL .EQ. 252) WRITE (2, 1252)
1252 FORMAT (' IGET NOT EQUAL TO 0 OR 1' )
     IF (IFAIL .GT. 252) WRITE (2, 5000)
     RETURN
     END
```

# APPENDIX

## ALGORITHMIC COMPLEXITY: THREE NP-HARD PROBLEMS IN COMPUTATIONAL STATISTICS

## 1. INTRODUCTION

Consider the following scenario. An algorithm has been devised to solve a computational problem, for definiteness say clustering of entities with some optimality criterion, and a program has been implemented which we assume is correct. The algorithm has successfully solved several test examples but, when presented with a more realistic, practical-sized case with many entities, the program will not complete execution within an acceptable amount of computer time. This undesirable situation may be attributed to an inefficient algorithm, an inherently difficult computational problem, or both. In the first case other algorithms might be considered, whereas in the second one is warned that the problem is intrinsically complex and a reformulation may be necessary.

In this paper we shall be concerned with the time-complexity of algorithms to solve certain problems: that is, the relationship between the size of an instance of the generic problem and the time required to solve it. By employing the theory of NP-completeness and NP-hardness, reviewed in Section 2 since it is probably not well known to statisticians, we may identify certain computational problems as members of a class thought to be intractable. Three computational problems in statistics, namely optimal clustering, optimal subset selection in regression and D-optimal exact design of experiments, are shown in Sections 3, 4 and 5 to be members of this class.

## 2. NP-COMPLETENESS AND NP-HARDNESS

We now give some definitions and a brief review of the essential concepts of the theory of NP-completeness and NP-hardness. For a comprehensive review the reader is referred to Garey and Johnson (1979), whereas for a more concise account see Tarjan (1978).

The theory is directed primarily towards yes/no decision problems usually possessing some parameters to which values are assigned to define instances of the generic problem. The problems of interest to the statistician, whilst often not of this form, usually have decision

analogues. For example, consider the following problem, to be examined further in Section 4.

OPTIMAL SUBSET SELECTION IN REGRESSION. The parameters are three positive integers n, k and q ($q \leq k$), an nxl data vector $\underline{Y}$ and an nxk design matrix $\underline{X}$ and the objective is to find a kxl vector $\hat{\underline{\beta}}$ with only q nonzero entries such that the residual sum of squares $R(\hat{\underline{\beta}}) = (\underline{Y} - \underline{X}\hat{\underline{\beta}})^T(\underline{Y} - \underline{X}\hat{\underline{\beta}})$ is minimized. An instance is defined by giving values to n, k, q, $\underline{Y}$ and $\underline{X}$.
The yes/no decision analogue may be formed in this case by adding a further parameter B to those above and, instead of minimizing $R(\hat{\underline{\beta}})$, asking whether there exists a kxl vector $\hat{\underline{\beta}}$ with only q nonzero entries such that $R(\hat{\underline{\beta}}) \leq B$.

The generic problem of interest will typically have many possible instances and we are concerned with the relationship between the "size" of an instance and the time required for execution. We only use an informal notion of the size s(I) of an instance I: the input length presented to the computer by a reasonable encoding of I. Similarly, the computer model need not concern us. The time-complexity function T(n) of an algorithm solving a given generic problem is defined as

$$T(n) = \max_{I:s(I)=n} \{\text{time required to solve I}\}$$

and we note that this definition is based upon the worst-time case.

Edmonds (1965) emphasized the important distinction between polynomial time algorithms, with T(n) bounded above by a polynomial, and exponential algorithms, where such a bound does not exist. Problems solvable by a polynomial time algorithm constitute the class P, whereas those only solvable in exponential time have been termed intractable as exponential algorithms quickly require such vast amounts of computer time to execute as n increases that, frequently, even moderate-sized instances are infeasible. The theory is concerned with identifying intractability.

In order to provide a general, mathematically convenient algorithmic

framework, sufficiently powerful to include exponential time, Cook (1971) introduced the idea of a nondeterministic polynomial time algorithm: a nondeterministic "guessing" stage repeatedly guesses structures and a polynomial time checking stage verifies whether the proposed structure proves the answer to be "yes" or not. The number of guesses need not be polynomially bounded and many complex problems may be solved by such an algorithm using an exponential number of guesses. These algorithms are not intended as a practical means of solution, but are a conceptual device for defining the class NP of problems that can be solved by nondeterministic polynomial time algorithms. For our subset selection in regression decision analogue, the guessing stage might choose a subset of q elements of $\hat{\underline{\beta}}$ allowed to take nonzero values and the checking stage would elucidate whether the least squares fit achieves the bound B.

Clearly, P is contained in NP, since a problem solvable in polynomial time could have a nondeterministic polynomial algorithm with a null guessing stage and a polynomial verification stage providing the solution. However, though intuitively reasonable, it remains to be proved that the complement of P in NP, NP-P, is not empty: not one of the very extensive list of problems in NP has been shown to be outside P and such a proof seems remote. Hence, Cook's weaker theory attempts to prove that a problem is "as hard" as any other in NP and, therefore, if NP-P is not empty (as is widely believed) it is as hard as those belonging to NP-P and also has no polynomial time algorithm. More precisely, a problem $\Pi$ is NP-complete if (i) $\Pi \in$ NP and (ii) any other problem $\Pi' \in$ NP is transformable to $\Pi$ in polynomial time; that is, given any instance of $\Pi'$ there exists a polynomial time algorithm (for instances of $\Pi'$) that will transform it into an instance of $\Pi$ such that the answer to $\Pi'$ is "yes" if and only if the answer to $\Pi$ is "yes". If $\Pi'$ is polynomially transformable to $\Pi$ and no polynomial time algorithm exists for $\Pi'$ then $\Pi$ must also be intractable since, otherwise, the transformation would provide a method of solving $\Pi'$ in polynomial time. It is in this sense that $\Pi$ is at least as hard as $\Pi'$, the NP-complete problems are the hardest in NP, and all NP-complete problems are intractable if any one is (NP-P is not empty).

As polynomial transformability is transitive, given one problem $\Pi'$ known to be NP-complete a simpler method of showing $\Pi$ to be also NP-complete is to prove that $\Pi \in$ NP and $\Pi'$ is polynomially transformable to $\Pi$. Cook supplied the first NP-complete problem, Satisfiability, and Karp (1972) added some practical combinatorial problems by transformation.

We may generalize the idea of NP-completeness to certain problems outside NP. If a problem $\Pi$, not necessarily in NP, is such that any instance of an NP-complete problem $\Pi'$ can be polynomially transformed into an instance of $\Pi$ and the solution of $\Pi$ (now not necessarily of the yes/no type) determines the answer to $\Pi'$, then $\Pi$ is at least as hard as the NP-complete problems, $\Pi$ cannot be solved in polynomial time unless NP-P is empty, and $\Pi$ is termed NP-hard. Rather than show that the decision analogues of the examples from statistical computation are NP-complete, we now prove that the original optimality problems are NP-hard.

## 3. CLUSTER ANALYSIS

The objective of cluster analysis is to partition a given set of n entities into a specified number, K, of initially undefined, mutually exclusive and exhaustive groups or clusters according to some optimality criterion. We only consider the situation where the available information comprises "distances", not necessarily Euclidean, and the criterion reflects the desire to achieve homogeneous, well separated clusters in this metric. Possible criteria include

(i) minimize the maximum within cluster distance,
(ii) minimize the sum of average within cluster squared distances, and
(iii) minimize the total within cluster distance.

For reviews of cluster analysis see Cormack (1971) and Everitt (1974).

As the number of possible partitions is approximately $K^n$, the computational requirements of a naive examination of all partitions are usually prohibitive and most algorithms find good but not necessarily

optimal solutions [see, for example, Friedman and Rubin (1967)]. Hansen and Delattre (1978) have shown that the decision problem analogue of the optimality problem with criterion (i) is NP-complete and, further, give a branch and bound algorithm which greatly reduces the number of partitions requiring examination to guarantee an exact solution. We extend the Hansen and Delattre result to show that optimal clustering is NP-hard for a much wider class of criteria, including (i), (ii) and (iii) above. Let a partition $P_K$ into K clusters be called perfect if all within cluster distances are zero (note, however, that the clusters need not be well separated as no conditions are imposed on the between cluster distances). Consider the class of criteria minimize $d(P_K)$, where $d(P_K)$ is a function of the within cluster distances such that $d(P_K) \geq 0$ with $d(P_K) = 0$ if and only if $P_K$ is perfect.

Theorem 1. Optimal clustering is NP-hard for any criterion minimize $d(P_K)$ such that $d(P_K) \geq 0$ and $d(P_K) = 0$ if and only if $P_K$ is a perfect partition.

Proof. Following Hansen and Delattre we polynomially transform one of Karp's NP-complete problems: Graph K-Colourability (GKC), also known as chromatic number. An instance of GKC is provided by a graph $G = (V, E)$ and a positive integer K and the question posed asks whether G is K-colourable; that is, does there exist a function f mapping every vertex $v \in V$ onto the integers $\{1, \ldots, K\}$ such that $f(u) \neq f(v)$ whenever the edge $\{u, v\} \in E$.

Take an instance of GKC. The n vertices of G will correspond to entities and the presence or absence of edges between vertices to distances. We define the distance $d(u, v)$ between entities u and v as $d(u, v) = 1$ if $\{u, v\} \in E$ and 0 otherwise. As there are only $\frac{1}{2} n (n - 1)$ distances to compute the transformation can be achieved in polynomial time. Now consider the solution of the optimal clustering problem with K clusters: if minimum $d(P_K) = 0$ the perfect partition found demonstrates that G is K-colourable whereas if minimum $d(P_K) > 0$ then no perfect partition exists and G is not K-colourable. Hence, optimal clustering for the defined class of criteria is NP-hard and no polynomial time algorithm

exists unless NP-P is empty.

## 4. SUBSET SELECTION IN REGRESSION ANALYSIS

Optimal subset selection in regression analysis is a well known problem in computational statistics reviewed by Hocking (1976). For a definition of the computational problem see Section 2. In practice, we might be interested in trying various values of q, the parameter defining the subset size, but this leaves the problem essentially unchanged for our purposes.

As the possible subsets number $\binom{k}{q}$ and increase exponentially with k, efficient algorithms have attempted to find an optimal subset by employing branch and bound to explore only a fraction of the total. Algorithms suitable for most practical examples are described by, for example, Beale, Kendall and Mann (1967), Lamotte and Hocking (1970) and Furnival and Wilson (1974). That these methods can cope with realistic examples is a reflection of the relatively small values of k typically found (less than 50 say) since the optimality problem is, in fact, NP-hard as we demonstrate below.

Theorem 2. Optimal subset selection in regression is NP-hard.

Proof. We polynomially transform Minimum Weight Solution to Linear Equations (MWSTLE), shown by Garey and Johnson to be NP-complete. An instance comprises an nxk matrix $\underline{X}$, an nxl vector $\underline{Y}$ and a positive integer $q \leq k$ and we ask whether there exists a kxl vector $\hat{\underline{\beta}}$ with at most q non-zero entries such that $\underline{X}\hat{\underline{\beta}} = \underline{Y}$. The transformation is trivial: take the parameters n, k, q, $\underline{X}$, $\underline{Y}$ of any instance of MWSTLE and solve the same optimal subset selection in regression instance. If we achieve $R(\hat{\underline{\beta}}) = 0$, then $\underline{X}\hat{\underline{\beta}} = \underline{Y}$ and the answer to the instance of MWSTLE is "yes", whereas if $R(\hat{\underline{\beta}}) > 0$ then the answer is "no". Therefore, optimal subset selection in regression is NP-hard.

## 5. D-OPTIMAL EXACT DESIGN OF EXPERIMENTS

Here we are concerned with choosing n experimental settings or design points $x_{(1)}, \ldots, x_{(n)}$, not necessarily distinct and possibly

vector-valued, from a design space $\mathcal{X}$ of r candidate sites $x_1, \ldots, x_r$. At the design points we shall observe yields $Y_1, \ldots, Y_n$ according to the linear model

$$Y_i = \underline{f}(x_{(i)})^T \underline{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n$$

or, in matrix notation,

$$\underline{Y} = \underline{X}\,\underline{\beta} + \underline{\varepsilon}.$$

The matrix $\underline{X}$ is an nxk design matrix ($k \leq n$) with row i containing $\underline{f}(x_{(i)})^T$ and $\underline{f}$ is a kxl vector of linearly independent functions on $\mathcal{X}$, $\underline{\beta}$ is a kxl vector of unknown parameters, and $\underline{\varepsilon}$ is an nxl vector of independently distributed errors, each with mean zero and constant variance. A D-optimal design is a choice of the n design points such that $\det(\underline{X}^T\underline{X})$ is maximized, whereupon the generalized variance of the least squares estimates of $\underline{\beta}$ is minimized (see St. John and Draper (1975) for a review of D-optimality). If resource constraints fix n at a small value, as we assume, such designs are called exact and the computationally simple, approximate design theory of Kiefer and Wolfowitz (1959) is inappropriate.

There are $^{r+n-1}C_n$ possible designs over which to maximize $\det(\underline{X}^T\underline{X})$ and most algorithms have only attempted to find good but not necessarily optimal solutions; the DETMAX algorithm of Mitchell (1974) is probably the most successful. More recently, Welch (1981) describes a branch and bound algorithm guaranteeing an optimal solution, but limited to moderate-sized problems (r, $n < 30$, say). We now show that the optimality problem is NP-hard.

Theorem 3. D-optimal exact design of experiments is NP-hard.

Proof. The target problem for polynomial transformation is another proved by Karp to be NP-complete, namely Hamiltonian Circuit (HC). An instance of HC is specified by a graph G=(V, E) with k vertices and r edges and we ask whether G contains a Hamiltonian circuit, that is, an ordering $v_{\Pi(1)}, \ldots, v_{\Pi(k)}$ of the vertices $v_1, \ldots, v_k$ of G such that $\{v_{\Pi(k)}, v_{\Pi(1)}\} \in E$ and $\{v_{\Pi(i)}, v_{\Pi(i+1)}\} \in E$ $(1 \leq i < k)$.

In the corresponding experiment the k vertices are associated with k unknown treatment parameters $\beta_1, \ldots, \beta_k$ and we may only make observations on the differences between certain pairs of parameters, those allowed being determined by the edges E. In fact, we define the r candidate sites as the set $\mathcal{X}$ where $\underline{x}_{ij} \in \mathcal{X}$ if and only if $\{v_i, v_j\} \in E$ and $\underline{x}_{ij}$ is a kx1 vector of zeros except for +1 in position i and -1 in position j (the ordering of i and j is unimportant) and we are limited to taking only k observations. The linear model

$$Y_i = \underline{x}_{(i)}^T \underline{\beta} + \varepsilon_i, \quad i = 1, \ldots, k$$

would be over-parameterized since we can only make observations on the differences between parameters, but one reparametrization is

$$Y_i = \underline{f}(\underline{x}_{(i)})^T \underline{\gamma} + \varepsilon_i, \quad i = 1, \ldots, k$$

where $\underline{f}(\underline{x}_{(i)})$ is a (k-1)x1 vector of the first k-1 elements of $\underline{x}_{(i)}$ and $\underline{\gamma}$ is the (k-1)x1 vector $(\beta_1 - \beta_k, \ldots, \beta_{k-1} - \beta_k)$. The matrix $\underline{X}$ is therefore kx(k-1) with row i containing $\underline{f}(\underline{x}_{(i)})^T$. We now show that max det $(\underline{X}^T\underline{X}) = k$ if and only if G contains a Hamiltonian circuit.

If G does contain a Hamiltonian circuit $v_{\pi(1)}, \ldots, v_{\pi(k)}$ then the points $\underline{x}_{\pi(k)\pi(1)}$ and $\underline{x}_{\pi(i)\pi(i+1)}$ $(1 \leq i < k)$ belong to $\mathcal{X}$ and they may be chosen as the k design points. By considering the structure imposed on the $\underline{X}$ matrix by such a choice we find that det $(\underline{X}^T\underline{X}) = k$. Conversely, suppose det $(\underline{X}^T\underline{X}) = k$ for a design matrix $\underline{X}$. Now

$$\det(\underline{X}^T\underline{X}) = \sum_{i=1}^{k} \det(\underline{X}_{-i}^T \underline{X}_{-i}) = \sum_{i=1}^{k} \left[\det(\underline{X}_{-i})\right]^2$$

where $\underline{X}_{-i}$ is the (k-1)x(k-1) matrix formed by deleting row i from $\underline{X}$. The application of row and column operations not changing det $(\underline{X}_{-i})$ demonstrates that det $(\underline{X}_{-i}) = \pm 1$ for any nonsingular $\underline{X}_{-i}$ and, therefore, det $(\underline{X}^T\underline{X}) = k$ implies that $\underline{X}_{-i}$ is nonsingular $(1 \leq i \leq k)$. Three necessary conditions for all $\underline{X}_{-i}$ to be nonsingular $(1 \leq i \leq k)$ are:

(i) all rows of $\underline{X}_{-i}$ are distinct and hence all rows of $\underline{X}$ are distinct;

(ii) no column of $\underline{X}_{-i}$ consists entirely of zeros and hence $\underline{X}$ must include at least two rows with $\pm 1$ in column j $(1 \leq j < k)$ (if there were only one such row it would be deleted for one $\underline{X}_{-i}$), and

(iii) for at least one row of $\underline{X}_{-i}$ the row sum is not zero and, hence, at least two row sums of $\underline{X}$ must be nonzero.

Therefore, if det $(\underline{X}^T\underline{X}) = k$, from (i) all design points must be distinct, by (ii) they include at least two comparisons of treatment j with another $(1 \leq j < k)$ and by (iii) they must also include at least two comparisons of treatment k with another (only such design points have a nonzero row sum in $\underline{X}$). The design points must, therefore, correspond to edges forming a Hamiltonian circuit of the graph G. Therefore, G contains a Hamiltonian circuit if and only if max det $(X^TX) = k$ and D-optimal exact design of experiments is NP-hard.

## 6. CONCLUSIONS

We have proved that three optimality problems found in computational statistics are NP-hard; that is, unless NP-P is empty they cannot be solved in polynomial time. However, the reader may be perturbed by the use of transformations in all three proofs that led to a mapping of instances of the target problem into instances of the statistical problem that are pathological or at least not typical of those commonly encountered. There is, indeed, some truth in this argument since the worst-case definition of time-complexity employed here and elsewhere would allow an atypical, small subset of the optimality problem to dominate the time-complexity function. A redefinition of time-complexity in terms of an average case would lead to great difficulties, though, firstly in specifying a probability distribution over instances and, secondly, in proving intractibility. Unfortunately, this is the state of the art.

On the other hand, some practical benefits can ensue from such results: if an optimality problem is known to be NP-hard we are warned that inherently complex cases can arise within the general framework, so prompting a re-examination of the problem, perhaps limiting the objective of an algorithm to finding good but not necessarily optimal solutions adequate for most practical purposes. If an exact solution is required to an optimality problem known to be NP-hard then, as in the three examples cited here, efficient approaches usually involve branch and bound or some other partial enumeration technique.

REFERENCES

BEALE, E.M.L., KENDALL, M.G. and MANN, D.W. (1967). The discarding of variables in multivariate analysis. Biometrika, 54, 357-366.

COOK, S.A. (1971). The complexity of theorem-proving procedures. Proc. Third Annual ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 151-158.

CORMACK, R.M. (1971). A review of classification. J. Roy. Statist. Soc. Ser. A, 134, 321-367.

EDMONDS, J. (1965). Paths, trees, and flowers. Canad. J. Math., 17, 449-467.

EVERITT, B.S. (1974). Cluster analysis. Heinemann, London.

FRIEDMAN, H.P. and RUBIN, J. (1967). On some invariant criteria for grouping data. J. Amer. Statist. Assoc., 62, 1159-1178.

FURNIVAL, G.M. and WILSON, R.W. (1974). Regressions by leaps and bounds. Technometrics, 16, 499-511.

GAREY, M.R. and JOHNSON, D.S. (1979). Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco.

HANSEN, P. and DELATTRE, M. (1978). Complete-link cluster analysis by graph coloring. J. Amer. Statist. Assoc., 73, 397-403.

HOCKING, R.R. (1976). The analysis and selection of variables in linear regression. Biometrics, 32, 1-49.

KARP, R.M. (1972). Reducibility among combinatorial problems, in R.E. Miller and J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 85-103.

KIEFER,J. and WOLFOWITZ, J. (1959). Optimum designs in regression problems. Ann. Math. Statist., 30, 271-294.

LAMOTTE, L.R. and HOCKING, R.R. (1970). Computational efficiency in the selection of regression variables. Technometrics, 12, 83-93.

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

ST. JOHN, R.C. and DRAPER, N.R. (1975). D-optimality for regression designs: a review. Technometrics, 17, 15-23.

TARJAN, R.E. (1978). Complexity of combinatoral algorithms. SIAM Rev., 20, 457-491.

WELCH, W.J. (1981). Branch and bound search for experimental designs based upon D-optimality and other criteria. Submitted for publication.