Chapter 24
Details of the OPTEX Procedure

Chapter Table of Contents

Chapter 24
Details of the OPTEX Procedure

Syntax

You can specify the following statements with the OPTEX procedure. Items within the brackets <> are optional.

PROC OPTEX < *options* > **;**
  CLASS *class-variables* **;**
  MODEL *effects* < / *options*> **;**
  BLOCKS *block-specification* <*options*> **;**
  EXAMINE <*options*> **;**
  GENERATE <*options*> **;**
  ID *variables* **;**
  OUTPUT OUT= *SAS-data-set* <*options*> **;**

To generate a design, you use the PROC OPTEX and MODEL statements. You can use the other statements as needed. The OPTEX procedure is interactive and allows you to use all statements (except the PROC OPTEX statement) after the first RUN statement.

Statement Ordering for Covariate Designs

You use the CLASS and MODEL statements to define a linear model for the runs in the candidate data set. You can also use these statements to define a general covariate model. In this case, list the CLASS and MODEL statements that define the model for the candidate points directly after the PROC OPTEX statement. Then list the CLASS and MODEL statements that define the covariate model after the BLOCKS DESIGN= specification. Thus, in this case, the ordering for these statements should be

1. PROC OPTEX statement
2. CLASS and MODEL statements for the candidate points
3. BLOCKS DESIGN= statement
4. CLASS and MODEL statements for the covariates

Note also that a CLASS statement naming classification variables must precede the MODEL statement that uses those variables.

Summary of Functions

Table 24.1, Table 24.2, and Table 24.3 classify the OPTEX statements and options by function.

Table 24.1. Summary of Options for Specifying the Design

Function	Statement	Option
Design Characteristics		
Number of design points	GENERATE	N=*number*
Saturated design	GENERATE	N=SATURATED
Augmented design	GENERATE	AUGMENT=*SAS-data-set*
Bayesian optimal design	MODEL	/ PRIOR=$p_1, p_2, \ldots$
Optimality Criteria		
Minimize trace of $(X'X)^{-1}$	GENERATE	CRITERION=A
Maximize $\|X'X\|$	GENERATE	CRITERION=D
Minimize mean minimum distance to design	GENERATE	CRITERION=U
Maximize mean distance between nearest design points	GENERATE	CRITERION=S
Model Specification		
Specify independent effects	MODEL	*effects*
Exclude intercept term	MODEL	*effects* NOINT
Specify class variables	CLASS	*variables*
Static coding	PROC OPTEX	CODING=STATIC
Orthogonal coding	PROC OPTEX	CODING=ORTH
Orthogonal coding with respect to candidates only	PROC OPTEX	CODING=ORTHCAN
Suppress coding of effects	PROC OPTEX	NOCODE
Block Specification		
Specify general covariance matrix for runs	BLOCKS	COVAR=*SAS-data-set* <*options*> VAR=*variables*
Specify general covariate model	BLOCKS	DESIGN=*SAS-data-set* <*options*>
Specify b blocks of size k	BLOCKS	STRUCTURE=$(b)k$ <*options*>
Options for block specifications		
Repeat the search n times		ITER=n
Retain best m searches		KEEP=m
Select initial design at random		INIT=RANDOM
Select initial design in order		INIT=CHAIN
Initial Design Characteristics		
Random and sequential methods	GENERATE	INITDESIGN=PARTIAL$< (m) >$
Random initial design	GENERATE	INITDESIGN=RANDOM
Sequential initial design	GENERATE	INITDESIGN=SEQUENTIAL

Function	**Statement**	**Option**
Specify initial design	GENERATE	INITDESIGN=*SAS-data-set*

Table 24.2. Summary of Options for Searching for the Design

Function	**Statement**	**Option**
Design Search Specification		
Retain best n searches	GENERATE	KEEP=n
Search n times	GENERATE	ITER=n
Specify candidate points	PROC OPTEX	DATA=*SAS-data-set*
Specify random seed	PROC OPTEX	SEED=*number*
Specify effective zero	PROC OPTEX	EPSILON=ϵ
Design Search Methods		
DETMAX algorithm with maximum excursion *level*	GENERATE	METHOD=DETMAX$<$*(level)*$>$
Exchange algorithm	GENERATE	METHOD=EXCHANGE
k-Exchange algorithm	GENERATE	METHOD=EXCHANGE$<(k)>$
Sequential algorithm	GENERATE	METHOD=SEQUENTIAL
Fedorov algorithm	GENERATE	METHOD=FEDOROV
Modified Fedorov algorithm	GENERATE	METHOD=M_FEDOROV

Table 24.3. Summary of Options for Examining and Saving the Design

Function	**Statement**	**Option**
Save the Design		
Best design	OUTPUT OUT=*SAS-data-set*	
Specific design	OUTPUT OUT=*SAS-data-set*	NUMBER=*design-number*
Block variable name	OUTPUT OUT=*SAS-data-set*	BLOCK=*variable-name*
Specify transfer variables	ID	*variables*
List the Design		
Design characteristics	EXAMINE	
Design points	EXAMINE	DESIGN
Information matrix $X'X$	EXAMINE	INFORMATION
Specific optimal design	EXAMINE	NUMBER=*design-number*
Variance matrix $(X'X)^{-1}$	EXAMINE	VARIANCE
Suppress all output	PROC OPTEX	NOPRINT

Statement Descriptions

This section provides detailed syntax information for the OPTEX procedure statements, beginning with the PROC OPTEX statement. The remaining statements are presented in alphabetical order.

PROC OPTEX Statement

PROC OPTEX <*options*> **;**

You use the PROC OPTEX statement to invoke the procedure. The following *options* can be used:

CODING=NONE
CODING=STATIC
CODING=ORTH
CODING=ORTHCAN

specifies which type of coding to use for modeling effects in the design. Coding equalizes all model effects as far as the optimization is concerned. The default is CODING=STATIC, which specifies that the values of all effects are to be coded to have maximum and minimum values of $+1$ and -1, respectively. The options CODING=ORTH and CODING=ORTHCAN specify orthogonal coding with respect to the input points. The option CODING=NONE suppresses coding of effects; it is equivalent to the NOCODE option. For more details on coding, see "Design Coding" on page 774.

Note that while CODING=STATIC is the default, CODING=ORTH will usually give more appropriate efficiency values, especially if all possible combinations of factor levels occur in the candidate data set.

DATA=*SAS-data-set*

specifies the input SAS data set that contains the candidate points for the design. By default, the OPTEX procedure uses the most recently created SAS data set. For details, see "DATA= Data Set" on page 768.

EPSILON=ϵ

specifies the smallest value ϵ that is considered to be nonzero for determining when the search is no longer yielding an improved design and when the information matrix for the design is singular. By default, ϵ = 0.00001.

NAMELEN=*n*

specifies the length of effect names in tables and output data sets to be n characters long, where n is a value between 20 and 200 characters. The default length is 20 characters.

NOCODE

suppresses the coding of effects in the model for the design. This option is equivalent to CODING=NONE.

NOPRINT

suppresses all output. This is useful when you only want the final design to be saved in a data set.

SEED=s

specifies a number s used to start the pseudo-random number generator (see “Search Methods” on page 780). The number s can be any positive integer up to $2^{31} - 1$. The default value of s is generated from the time of day.

STATUS=*status-level*

specifies that the status of the search be checked at the given level, where *status-level* is an integer between 1 and 4, inclusive. If you specify a *status-level* then a table of the status at each check point is displayed. You can use this table to track the progress of long searches. The allowable *status-levels* are listed in the following table:

Status-level	Checks status after each:
1	design search; the number of searches specified by the NITER= option
2	search loop
3	internal search loop
4	extra internal search loop for METHOD=M_FEDOROV

Each search method loops to produce successively better designs; these are the search loops for STATUS=2. STATUS=3 and STATUS=4 refer to deeper loops within the search methods. You will only need to specify STATUS=3 or STATUS=4 very rarely, since unless simply evaluating a potential switch is very expensive (as it can occasionally be with the space-filling criteria). Evaluating and displaying the status at this level will make the search much, much slower.

BLOCKS Statement

BLOCKS *block-specification* < *options* > **;**

You use the BLOCKS statement to find a D-optimal design in the presence of fixed covariates (for example, blocks) or covariance. The technique is an extension of the optimal blocking technique of Cook and Nachtsheim (1989); see “Optimal Blocking” on page 782.

For the purposes of optimal blocking, the model for the original candidate points is referred to as the *treatment model*; the candidate points for the part of the design matrix corresponding to the treatment model form the *treatment set*. If the GENERATE statement is not specified, then the full candidate set is used as the treatment set; otherwise, an optimal design for the treatment model ignoring the blocks is first generated, and the result is used as the treatment set for optimal blocking.

The following are three mutually exclusive *block-specifications* that you can provide:

COVAR=*SAS-data-set* **VAR=(** *variables* **)**

specifies a data set to use in providing a general covariance matrix for the runs. The argument to VAR= names the variables in this data set that contain the columns of the covariance matrix for the runs. For an example, see Example 24.9 on page 760.

DESIGN=*SAS-data-set*

specifies a data set to use in providing a general covariate model. In addition to this data set, you must specify a covariate model with the CLASS and

MODEL statements. Covariate models are specified in the same way as the treatment model; CLASS and MODEL statements that come after a BLOCKS statement involving the DESIGN= specification are interpreted as applying to the covariate model. For an example, see Example 24.8 on page 758.

STRUCTURE=(b) k

specifies a block design with b blocks of size k. For an example, see Example 24.7 on page 755.

The following *options* can also be used:

INIT=RANDOM

specifies the initialization method for constructing the starting design. The option INIT=RANDOM specifies that the starting design is to be constructed by selecting candidates at random without replacement. The option INIT=CHAIN selects candidate points in the order in which they occur in the original data set.

ITER=n

specifies the number of times to repeat the search from different initial designs. Because local optima are common in difficult search problems, it is often a good idea to make several tries for the optimal design with a random or partially random method of initialization (see the preceding INIT= option). By default, $n = 10$. You can specify ITER=0 to evaluate the initial design itself.

KEEP=m

specifies that only the best m designs are to be retained. The value m must be less than or equal to the value n of the ITER= option; by default $m = n$, so that all iterations are kept. This option is useful when you want to make many searches to overcome the problem of local optima but you are only interested in the results of the best m designs.

NOEXCHANGE

suppresses the part of the optimal blocking algorithm that exchanges treatment design points for candidate treatment points. When this option is specified, only interchanges between design points are performed. Use this option when you do not want to change which treatment points are included in the design and you only want to find their optimal ordering.

CLASS Statement

CLASS *class-variables* **;**

You use the CLASS statement to identify classification variables, which are factors that separate the observations into groups. For example, a completely randomized design has a single *class-variable* that identifies the groups of observations. A randomized complete block design has two *class-variables*; one identifies the blocks and one identifies the treatments.

Class-variables can be either numeric or character. The OPTEX procedure uses the formatted values of *class-variables* in forming model effects. Any variable in the model that is not listed in the CLASS statement is assumed to be continuous. Continuous variables must be numeric.

Note: If you specify a data set containing fixed covariate effects with a DESIGN= data set in the BLOCKS statement, then a CLASS or MODEL statement that follows the BLOCKS statement refers to the model for the fixed covariates. A CLASS or MODEL statement that defines the model for the candidate points (treatment model) should be specified *before* the BLOCKS statement.

EXAMINE Statement

EXAMINE <*options*> **;**

You use the EXAMINE statement to display the characteristics of a selected design. By default, the EXAMINE statement lists certain measures of design efficiency for the best design. (See the "Output" section on page 783.) The following *options* can be used to modify the output:

DESIGN
lists the actual points in the selected design.

INFORMATION
INFO
I
lists the information matrix $X'X$ for the selected design.

NUMBER=*design-number*
selects a design to examine by specifying its *design-number*. Designs are ordered by the value of the efficiency criterion that is being optimized. Thus, a *design-number* of 1 corresponds to the best design found, a *design-number* of 2 corresponds to the second best design, and so on. The default *design-number* is 1. To modify the number of designs created, see the ITER= option on page 740.

VARIANCE
VAR
V
lists the variance matrix $(X'X)^{-1}$ for the parameter estimates for the selected design.

For details on design efficiencies, see "Design Efficiency Measures" on page 773.

If you use the OPTEX procedure interactively, you must enter the options for every EXAMINE statement. For example, the following statements list default information and the design points for the best design but only default information for the second-best design:

```
examine number=1 design;
examine number=2;
```

The following statements list default information and design points for both the best and second-best designs:

```
examine number=1 design;
examine number=2 design;
```

GENERATE Statement

GENERATE <*options*> **;**

You use the GENERATE statement to customize the search for a design. By default, the OPTEX procedure searches for a design as follows:

- using the exchange algorithm (METHOD=EXCHANGE)
- using D-optimality as the optimality criterion (CRITERION=D)
- using a completely random initial design to start the search (INITDESIGN=RANDOM)
- selecting candidate points only from the DATA= data set (modified by using AUGMENT= or INITDESIGN= data sets)
- performing 10 iterations in the search (ITER=10)
- finding a design with $10 + p$ points, where p is the number of parameters in the model (modified by using the N= or INITDESIGN= option)

The following *options* can be used to modify these defaults:

AUGMENT=*SAS-data-set*
specifies a data set that contains a design to be augmented, in other words, a set of points that must be contained in the design generated. When creating designs, the OPTEX procedure adds points from the DATA= data set (or the last data set created, if DATA= is not specified) to points from the AUGMENT= data set. The number of points in the design to be augmented must be less than the number of points specified with the N= option. For details, see “AUGMENT= Data Set” on page 768.

CRITERION=*crit*
specifies the optimality criterion used in the search. You can specify any one of the following:

CRITERION=D
specifies D-optimality; the optimal design maximizes the determinant $|X'X|$ of the information matrix for the design. This is the default criterion.

CRITERION=A
specifies A-optimality; the optimal design minimizes the sum of the variances of the estimated parameters for the model, which is the same as minimizing the trace of $(X'X)^{-1}$.

CRITERION=U
specifies U-optimality; the optimal design minimizes the sum of the minimum distances from each candidate point to the design. That is, if $\mathcal{C}$ is the set of candidate points, $\mathcal{D}$ is the set of design points, and $d(\mathbf{x}, \mathcal{D})$ is the minimum distance from $\mathbf{x}$ to any point in $\mathcal{D}$, then a U-optimal design minimizes

$$\sum_{\mathbf{x} \in \mathcal{C}} d(\mathbf{x}, \mathcal{D})$$

This measures how well the design "covers" the candidate set; thus, a U-optimal design is also called a *uniform coverage design*.

CRITERION=S

specifies S-optimality; the optimal design maximizes the harmonic mean of the minimum distance from each design point to any other design point. Mathematically, an S-optimal design maximizes

$$\frac{N_D}{\sum_{\mathbf{y}\in\mathcal{D}} 1/d(\mathbf{y},\mathcal{D}-\mathbf{y})}$$

where $\mathcal{D}$ is the set of design points, and N_D is the number of points in $\mathcal{D}$. This measures how spread out the design points are; thus, an S-optimal design is also called a *maximum spread design*.

For more information on the different criteria, see "Optimality Criteria" on page 776.

INITDESIGN=*initialization-method*

specifies a method of obtaining an initial design for the search procedure. Valid values of *initialization-method* are as follows:

SEQUENTIAL

specifies an initial design chosen by a sequential search. The design given by INITDESIGN=SEQUENTIAL is the same as the design given by METHOD=SEQUENTIAL. You can use the INITDESIGN=SEQUENTIAL option with other values of the METHOD= option to specify a sequential design as the initial design for various search methods. For details, see "Search Methods" on page 780.

RANDOM

specifies a completely random initial design. The initial design generated consists of a random selection of observations from the DATA= data set.

PARTIAL<**(***m* **)**>

specifies an initial design using a mixture of RANDOM and SEQUENTIAL methods. A small number n_r of points for the initial design are chosen at random from the candidates, and the rest of the design points are chosen by a sequential search. (For a definition of the sequential search, see "Search Methods" on page 780.)

By default, n_r is randomly chosen between 0 and half the number of parameters in the linear model. You can specify the optional integer m to modify the selection of n_r. If $m > 0$, then n_r is randomly chosen between 0 and m for each try. If $m < 0$, then $n_r = |m|$ for each try. The maximum value for $|m|$ is the number of points in the design. Refer to Galil and Kiefer (1980) for notes on choosing n_r.

SAS-data-set

specifies a data set that holds the initial design. Use this *initialization-method* when you have a specific design that you want to improve or when you want to evaluate an existing design. For details, see "INITDESIGN= Data Set" on page 768.

The default initialization method depends on the search procedure as shown in Table 24.4.

Table 24.4. Default Initialization Methods

Search Procedure (METHOD= option)	**Default Initialization Method** (INITDESIGN= option)
DETMAX	PARTIAL
EXCHANGE	RANDOM
FEDOROV	RANDOM
M_FEDOROV	PARTIAL
SEQUENTIAL	none

If you specify INITDESIGN=*SAS-data-set* and METHOD=SEQUENTIAL, no search is performed; the INITDESIGN= data set is taken as the final design. By specifying these options, you can use the procedure to evaluate an existing design.

ITER=n

specifies the number n of searches to make. Because local optima are common in difficult search problems, it is often a good idea to make several tries for the optimal design with a random or partially random method of initialization (see the preceding INITDESIGN= option). By default, $n = 10$.

The n designs found are sorted by their respective efficiencies according to the current optimality criterion (see the CRITERION= option on page 738.) The most efficient design is assigned a *design-number* of 1, the second most efficient design is assigned a *design-number* of 2, and so on. You can then use the *design-number* in the EXAMINE and OUTPUT statements to display the characteristics of a design or to save a design in a data set.

KEEP=m

specifies that only the best m designs are to be retained. The value m must be less than or equal to the value n of the ITER= option; by default $m = n$, so that all iterations are kept. This option is useful when you want to make many searches to overcome the problem of local optima but are interested only in the results of the best m designs.

METHOD=DETMAX<*(level)*>
METHOD=EXCHANGE$< (k) >$
METHOD=FEDOROV
METHOD=M_FEDOROV
METHOD=SEQUENTIAL

specifies the procedure used to search for the optimal design. The default is METHOD=EXCHANGE.

With METHOD=DETMAX, the optional *level* gives the maximum excursion level for the search, where *level* is an integer greater than or equal to 1. Enclose the value of *level* in parentheses immediately following the word DETMAX. The default value for *level* is 4. In general, larger values of *level* result in longer search times.

When METHOD=EXCHANGE, the optional k specifies the k-exchange search method of Johnson and Nachtsheim (1983), which generalizes the modified Fedorov search algorithm of Cook and Nachtsheim (1980). Enclose the value of k in parentheses immediately following the word EXCHANGE.

From fastest to slowest, the methods are

$$\text{SEQUENTIAL} \rightarrow \text{EXCHANGE} \rightarrow \text{DETMAX} \rightarrow \text{M\_FEDOROV} \rightarrow \text{FEDOROV}$$

In general, slower methods result in more efficient designs. While the default method EXCHANGE always works relatively quickly, you may want to specify a more reliable method, such as M_FEDOROV, with fast computers or small- to moderately-sized problems.

See “Search Methods” on page 780 for details on the algorithms.

N=n
N=SATURATED
specifies the number of points in the final design. The default design size is $10 + p$, where p is the number of parameters in the model. If you use the INITDESIGN= option, the default number is the number of points in the initial design. Specify N=n to search for a design with n points. Specify N=SATURATED to search for a design with the same number of points as there are parameters in the model. A saturated design has no degrees of freedom to estimate error and should be used with caution.

ID Statement

ID *variables* **;**

You use the ID statement to name the *variables* in the DATA= data set that are not involved in the model but are to be transferred from the input data set to the output data set.

Variables listed in the ID statement must be contained in the DATA= data set. They can also be contained in other input data sets. If an ID variable is also contained in an AUGMENT= or INITDESIGN= data set and an observation from that data set is used in the final design, the values of the ID variables for that observation are transferred to the OUT= data set. For details, see “Input Data Sets” on page 767.

MODEL Statement

MODEL *effects* < / *options* > **;**

You use the MODEL statement to specify the independent effects used to model data that are to be collected with the design that is being constructed. The *effects* can be

- simple continuous regressor effects
- polynomial continuous effects
- main effects of classification variables
- interactions of classification variables
- continuous-by-class effects

The variables used to form *effects* in the MODEL statement must be present in all input data sets. For details on input data sets, see “Input Data Sets” on page 767. For details on the specification of different types of effects and on how the design matrix is defined with respect to the effects, see “Specifying Effects in MODEL Statements” on page 770.

If you specify a data set containing fixed covariate effects with a DESIGN= data set in the BLOCKS statement, then a CLASS or MODEL statement that *follows* the BLOCKS statement refers to the model for the fixed covariates. A CLASS or MODEL statement that defines the model for the candidate points (treatment model) should occur *before* the BLOCKS statement.

The following options can be used in the MODEL statement:

NOINT
excludes the intercept parameter from the model. By default, the OPTEX procedure includes the intercept parameter in the model.

PRIOR=*num-list*
specifies prior precision values corresponding to groups of effects in the model. Groups of effects in the MODEL statement with the same prior precision must be separated by commas. Then use the PRIOR= option, listing as many prior precision values as there are groups of effects. See Example 24.6 on page 753 for an example.

When you specify prior precision values, the information matrix for estimating the linear parameters is $X'X + P$, where X is the design matrix and P is a diagonal matrix with the prior precision values that you specify on the diagonal. Thus, in terms of a prior distribution, the inverses of the prior precision values can be interpreted as prior variances for the linear parameters corresponding to each effect. As an alternative interpretation, note that with orthogonal coding the value of the prior for an effect says roughly how many prior “observations’ worth” of information you have for that effect. See “Design Coding” on page 774 for details on orthogonal coding.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set* $<$ *options* $>$ **;**

You use the OUTPUT statement to save a design in an output data set. By default, the saved design is the best design found. You specify the data set name as follows:

OUT=*SAS-data-set*
gives a name for the output data set. The OUT= data set is required in the OUTPUT statement.

The following *options* can be used:

BLOCKNAME=*variable-name*
specifies the name to be given to the blocking variable in the output data set. The default name is BLOCK. You can use this *option* in conjunction with a STRUCTURE= option in the BLOCKS statement. See Example 24.7 on page 755 for an example.

NUMBER=*design-number*
selects a design to output by specifying its *design-number*. Designs are ordered by the value of the efficiency criterion that is being optimized. Thus, a *design-number* of 1 corresponds to the best design found, a *design-number* of 2 corresponds to the second best design, and so on. The default *design-number* is 1. To modify the number of designs created, see the ITER= option on page 740.

Alternatively, you can specify one of the following:

NUMBER=DBEST
selects the design that has the highest D-efficiency value.

NUMBER=ABEST
selects the design that has the highest A-efficiency value.

NUMBER=GBEST
selects the design that has the highest G-efficiency value.

NUMBER=VBEST
selects the design that has the minimum average standard error for prediction.

These options can be used to find designs that are efficient for more than one criterion For example, you can use the default CRITERION=D option in the GENERATE statement with the NUMBER=GBEST option in the OUTPUT statement to find the D-optimal design that has maximal G-efficiency. In fact, this is the best way to use the OPTEX procedure to find G-efficient designs; see "G- and I-optimality" on page 777 for more details.

Advanced Examples

Example 24.1. Nonstandard Linear Model

See OPTEX3 in the SAS/QC Sample Library

The following example is based on an example in Mitchell (1974a). An animal scientist wants to compare wildlife densities in four different habitats over a year. However, due to the cost of experimentation, only 12 observations can be made. The following model is postulated for the density $y_j(t)$ in habitat j during month t:

$$y_j(t) \quad = \quad \mu_j + \beta t + \sum_{i=1}^{4} a_i \cos(i\pi t/4) + \sum_{i=1}^{3} b_i \sin(i\pi t/4).$$

This model includes the habitat as a classification variable, the effect of time with an overall linear drift term βt, and cyclic behavior in the form of a Fourier series. There is no intercept term in the model.

The OPTEX procedure is used since there are no standard designs that cover this situation. The candidate set is the full factorial arrangement of four habitats by 12 months, which can be generated with a DATA step, as follows:

```
data a;
   drop theta pi;
   array c{4} c1-c4;
   array s{3} s1-s3;
   pi = arcos(-1);
   do habitat=1 to 4;
      do month=1 to 12;
         theta = pi * month / 4;
         do i=1 to 4; c{i} = cos(i*theta); end;
         do i=1 to 3; s{i} = sin(i*theta); end;
         output;
      end;
   end;
run;
```

Data set A contains the 48 candidate points and includes the cosine variables (C1, C2, C3, and C4) and sine variables (S1, S2, S3, S4). The following statements produce Output 24.1.1:

```
proc optex seed=193030034 data=a;
   class    habitat;
   model    habitat month c1-c4 s1-s3 / noint;
   generate n=12;
run;
```

Output 24.1.1. Sampling Wildlife Habitats Over Time

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	31.6103	19.7379	57.7350	1.3229
2	31.6103	19.7379	57.7350	1.3229
3	31.6103	19.3793	57.7350	1.3229
4	31.6103	19.2916	57.7350	1.3229
5	31.6103	19.2626	57.7350	1.3229
6	31.6103	19.0335	57.7350	1.3229
7	30.1304	14.8837	44.7214	1.4907
8	30.1304	14.2433	44.7214	1.5092
9	30.1304	13.1687	44.7214	1.5456
10	28.1616	9.8842	40.8248	1.7559

The best determinant (D-efficiency) was found in 6 out of the 10 tries. Thus, you can be confident that this is the best achievable determinant. Only the A-efficiency distinguishes among the designs listed in Output 24.1.1. The best design has an A-efficiency of 19.74%, whereas another design has the same D-efficiency but a slightly smaller A-efficiency of 19.03%, or about 96% relative A-efficiency. To explore the differences, you can save the designs in data sets and print them. Since the OPTEX procedure is interactive, you need to submit only the following statements (immediately after the preceding statements) to produce Output 24.1.2 and Output 24.1.3:

```
   output out=d1 number=1;
run;
   output out=d6 number=6;
run;

proc sort data=d1;
   by  month habitat;
proc print data=d1;
   var month habitat;
run;

proc sort data=d6;
   by  month habitat;
proc print data=d6;
   var month habitat;
run;
```

Output 24.1.2. The Best Design

Obs	month	habitat
1	1	3
2	2	2
3	3	4
4	4	1
5	5	4
6	6	1
7	7	2
8	8	3
9	9	4
10	10	1
11	11	2
12	12	3

Output 24.1.3. Design with Lower A-Efficiency

Obs	month	habitat
1	1	4
2	2	2
3	3	3
4	4	1
5	5	1
6	6	4
7	7	4
8	8	1
9	9	2
10	10	1
11	11	4
12	12	3

Note the structure of the best design in Output 24.1.2. One habitat is sampled in each month, each habitat is sampled three times, and the habitats are sampled in consecutive complete blocks. Even though the design in Output 24.1.3 is as D-efficient as the best, it has almost none of this structure; one habitat is sampled each month, but habitats are not sampled an equal number of times. This demonstrates the importance of choosing a final design on the basis of more than one criterion.

You can try searching for the A-optimal design directly. This takes more time but (with only 48 candidate points) is not too large a problem. The following statements produce Output 24.1.4:

```
proc optex seed=193030034 data=a;
   class    habitat;
   model    habitat month c1-c4 s1-s3 / noint;
   generate n=12 criterion=A;
run;
```

Output 24.1.4. Searching Directly for an A-efficient Design

```
                                                                    Average
                                                                   Prediction
Design                                                              Standard
Number     D-Efficiency     A-Efficiency     G-Efficiency       Error
------------------------------------------------------------------------
     1        31.6103          19.7379          57.7350          1.3229
     2        30.1304          17.8273          52.2233          1.3894
     3        30.1304          17.7943          52.2233          1.3944
     4        30.1304          17.6471          52.2233          1.4093
     5        28.1616          15.7055          44.7214          1.4860
     6        28.1616          14.5289          44.7214          1.5343
     7        28.1616          13.8603          39.2232          1.5811
     8        25.0891          11.6152          37.7964          1.8143
     9        25.0891          10.7563          37.7964          1.8143
    10        25.0891          10.5437          33.3333          1.8930
```

The best design found is no more A-efficient than the one found previously.

Example 24.2. Comparing DETMAX Algorithm to Sequential Algorithm

An automotive engineer wants to fit a quadratic model to fuel consumption data in order to find the values of the control variables that minimize fuel consumption (refer to Vance 1986). The three control variables and their possible settings are shown in the following table:

See OPTEX4 in the SAS/QC Sample Library

Variable	Values							
AF	15	16	17	18				
EGR	0.020	0.177	0.377	0.566	0.921	1.117		
SA	10	16	22	28	34	40	46	52

Rather than run all 192 ($4 \times 6 \times 8$) combinations of these factors, the engineer would like to see whether the total number of runs can be reduced to 50 in an optimal fashion.

Since the factors have different numbers of levels, you can use the PLAN procedure (refer to the *SAS/STAT User's Guide*) to generate the full factorial set to serve as a candidate data set for the OPTEX procedure.

```
proc plan;
   factors af=4 ordered egr=6 ordered sa=8 ordered
           / noprint;
   output out=a
          af  nvals=(15,16,17,18)
          egr nvals=(.020,.177,.377,.566,.921,1.117)
          sa  nvals=(10,16,22,28,34,40,46,52);
run;
```

The DETMAX algorithm of Mitchell (1974a) is very commonly used for computer-generated optimal design. Although it is not the default search method for the OPTEX procedure, you can specify that it be used with the METHOD=DETMAX option in the GENERATE statement. For example, the following statements produce Output 24.2.1.

```
proc optex data=a seed=61552;
   model af|egr|sa@2 af*af egr*egr sa*sa;
   generate n=50 method=detmax;
run;
```

Output 24.2.1. Efficiencies with DETMAX Algorithm

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.4922	24.8987	95.2281	0.4202
2	46.4864	24.8562	95.5744	0.4205
3	46.4797	24.8830	95.3137	0.4203
4	46.4635	25.6461	94.8125	0.4175
5	46.4495	24.5376	95.5559	0.4237
6	46.4459	25.0749	94.8536	0.4197
7	46.4428	24.5111	95.3704	0.4240
8	46.4333	25.0321	95.1371	0.4199
9	46.4333	25.0321	95.1371	0.4199
10	46.4333	25.0321	95.1371	0.4199

The DETMAX search method can require considerable run time. For comparison, you can use the METHOD=SEQUENTIAL option in the GENERATE statement, as shown in the following statements, which produce Output 24.2.2.

```
proc optex data=a seed=33805;
   model af|egr|sa@2 af*af egr*egr sa*sa;
   generate n=50 method=sequential;
run;
```

Output 24.2.2. Efficiencies with Sequential Algorithm

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.4009	25.0472	93.8673	0.4200

In a fraction of the run time required by DETMAX, the sequential algorithm finds a design with a relative D-efficiency of $46.4009/46.4922 = 99.8\%$ compared to the best design found by the DETMAX procedure and with *better* A-efficiency. As this demonstrates, if absolute D-optimality is not required, a faster, simpler search may be sufficient.

Example 24.3. Using an Initial Design to Search an Optimal Design

This example is a continuation of Example 24.2 on page 747.

See OPTEX4 in the SAS/QC Sample Library

You can customize the runs used to initialize the search in the OPTEX procedure. For example, you can use the INITDESIGN=SEQUENTIAL option to use an initial design chosen by the sequential search. Or you can place specific points in a data set and use the INITDESIGN=*SAS-data-set* option. In both cases, the search time can be significantly reduced, since the search only has to be done once. This example illustrates both of these options.

The previous example compared the results of the DETMAX and sequential search algorithms. You can use the design chosen by the sequential search as the *starting point* for the DETMAX algorithm. The following statements specify the DETMAX search method, replacing the default initialization method with the sequential search:

```
proc optex data=a seed=33805;
   model af|egr|sa@2 af*af egr*egr sa*sa;
   generate n=50 method=detmax initdesign=sequential;
run;
```

The results, which are displayed in Output 24.3.1, show an improvement over the sequential design itself (Output 24.2.2) but not over the DETMAX algorithm with the default initialization method (Output 24.2.1). Evidently the sequential design represents a local optimum that is not the global optimum, which is a common phenomenon in combinatorial optimization problems such as this one.

Output 24.3.1. Initializing with a Sequential Design

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.4333	25.0321	95.1371	0.4199

Prior knowledge of the design problem at hand may also provide a specific set of factor combinations to use as the initial design. For example, many D-optimal designs are composed of replications of the optimal saturated design—that is, the optimal design with exactly as many points as there are parameters to be estimated. In this case, there are 10 parameters in the model. Thus, you can find the optimal saturated design in 10 points, replicate it five times, and use the resulting design as an initial design, as follows:

```
proc optex data=a seed=33805;
    model af|egr|sa@@2
          af*af egr*egr sa*sa;
    generate n=saturated
             method=detmax;
    output out=b;
```

```
data c; set b; drop i;
   do i=1 to 5; output; end;

proc optex data=a seed=33805;
    model af|egr|sa@@2
          af*af egr*egr sa*sa;
    generate n=50
             method=detmax
             initdesign=c;
run;
```

The results are displayed in Output 24.3.2 and Output 24.3.3. The resulting design is 99.9% D-efficient and 98.3% A-efficient relative to the best design found by the straight-forward approach (Output 24.2.1), and it takes considerably less time to produce.

Output 24.3.2. Efficiencies for the Unreplicated Saturated Design

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	41.6990	24.8480	67.6907	0.9508
2	41.4931	22.2840	70.8532	0.9841
3	40.9248	20.7672	62.2177	1.0247
4	40.7447	21.6253	52.7537	1.0503
5	39.9563	20.1557	46.4244	1.0868
6	39.9287	19.5856	45.9023	1.0841
7	39.9287	19.5856	45.9023	1.0841
8	38.9078	13.5976	37.7964	1.2559
9	38.9078	13.5976	37.7964	1.2559
10	37.6832	12.5540	45.3315	1.3036

Output 24.3.3. Initializing with a Data Set

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.4388	24.4951	96.0717	0.4242

Example 24.4. Optimal Design Using an Augmented Best Design

See OPTEX4 in the SAS/QC Sample Library

This example is a continuation of Example 24.2 on page 747.

You can specify a set of points that you want included in the final design found by the OPTEX procedure, using the AUGMENT= option in the GENERATE statement to specify a data set that contains a design to be augmented.

In this case, you can try to speed up the search for a 50-run design by first finding an optimal 25-run design and then augmenting that design with another 25 runs, as shown in the following statements:

```
proc optex data=a seed=36926;
    model af|egr|sa@2 af*af egr*egr sa*sa;
    generate n=25 method=detmax;
    output out=b;
proc optex data=a seed=37034;
    model af|egr|sa@2 af*af egr*egr sa*sa;
    generate n=50 method=detmax augment=b;
run;
```

The result (see Output 24.4.1 and Output 24.4.2) is a design with almost 100% D-efficiency and A-efficiency relative to the best design found by the first attempt. However, this approach is not much faster than the original approach, since the run time for the DETMAX algorithm is essentially linear in the size of the design (see "Memory and Run-Time Considerations" on page 779.)

Output 24.4.1. Efficiencies for the 25-point Design to be Augmented

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.2975	26.0374	91.1822	0.5849
2	46.2171	25.9733	86.4608	0.5859
3	46.1720	25.9378	88.3293	0.5860
4	46.1374	25.9128	86.1895	0.5866
5	46.0808	22.6647	86.1502	0.6169
6	46.0620	24.7326	89.7179	0.6012
7	45.9992	25.4549	90.3330	0.5946
8	45.9630	24.7610	88.2701	0.5991
9	45.9627	25.5310	88.5737	0.5894
10	45.7994	24.5645	87.7544	0.6005

Output 24.4.2. Efficiencies for the Augmented 50-point Design

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.4957	25.0858	94.8160	0.4195
2	46.4773	25.0696	95.0646	0.4195
3	46.4684	24.5519	96.1259	0.4234
4	46.4676	24.5002	95.6830	0.4238
5	46.4587	25.0709	94.6650	0.4196
6	46.4555	24.8087	95.7768	0.4209
7	46.4471	24.5460	95.0073	0.4240
8	46.4373	25.0740	94.4640	0.4194
9	46.3899	25.0007	95.2162	0.4201
10	46.3662	24.4013	94.9539	0.4242

Example 24.5. Optimal Design Using a Small Candidate Set

See OPTEX4 in the SAS/QC Sample Library

This example is a continuation of Example 24.4 on page 750.

A well-chosen initial design can speed up the search procedure, as illustrated in Example 24.2 on page 747. Another way to speed up the search is to reduce the candidate set. The following statements generate the optimal design with a fast, sequential search and then use the FREQ procedure to examine the frequency of different factor levels in the final design:

```
proc optex data=a seed=33805 noprint;
   model af|egr|sa@2 af*af egr*egr sa*sa;
   generate n=50 method=sequential;
   output out=b;
proc freq;
   table af egr sa / nocum;
run;
```

Output 24.5.1. Factor Level Frequencies for Sequential Design

af	Frequency	Percent
15	19	38.00
16	6	12.00
17	6	12.00
18	19	38.00

egr	Frequency	Percent
0.02	20	40.00
0.566	9	18.00
1.117	21	42.00

sa	Frequency	Percent
10	19	38.00
28	6	12.00
34	5	10.00
52	20	40.00

From Output 24.5.1, it is evident that most of the factor values lie in the middle or at the extremes of their respective ranges. This suggests looking for an optimal design with a candidate set that includes only those points in which the factors have values in the middle or at the extremes of their respective ranges. The following statements illustrate this approach (see Output 24.5.2):

```
proc plan;
   factors af=4 ordered egr=4 ordered sa=4 ordered
           / noprint;
   output out=a af  nvals=(  15,  16,  17,   18)
                egr nvals=(.020,.377,.566,1.117)
                sa  nvals=(  10,  28,  34,   52);
proc optex seed=61552;
   model af|egr|sa@2 af*af egr*egr sa*sa;
   generate n=50 method=detmax;
run;
```

Output 24.5.2. Optimal Design Using a Smaller Candidate Set

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.5151	24.9003	96.7226	0.4442
2	46.4997	24.5549	96.1157	0.4478
3	46.4920	24.5530	95.9941	0.4480
4	46.4657	24.8653	95.5627	0.4446
5	46.4547	24.5071	96.0385	0.4481
6	46.4333	25.0321	95.1371	0.4448
7	46.4333	25.0321	95.1371	0.4448
8	46.4333	25.0321	95.1371	0.4448
9	46.3916	24.3617	95.0041	0.4489
10	46.3379	24.8695	94.3115	0.4458

Once again, the resulting design is almost as good as the best one derived by a straightforward search ($>$ 99.9% relative D-efficiency and $>$ 98.5% relative A-efficiency) and takes much less time to find. Moreover, designs with fewer factor levels can be much easier to implement.

See “Handling Many Variables” on page 727 for another example of reducing the candidate set for the optimal design search.

Example 24.6. Bayesian Optimal Design

See OPTEX7 in the SAS/QC Sample Library

Suppose you want a design in 20 runs for seven two-level factors. There are 29 terms in a full second-order model, so you will not be able to estimate all main effects and two-factor interactions. If the number of runs were a power of 2, a design of resolution 4 could be used to estimate all main effects free of the two-factor interactions, as well as to provide partial information on the interactions. However, when the number of runs is not a power of two, as in this case, DuMouchel and Jones (1994) suggest searching for a *Bayesian optimal design* by specifying nonzero prior precision values for the interactions. You can specify these values in the OPTEX procedure with the PRIOR= option in the MODEL statement. This says that you want to consider all main effects and interactions as potential effects, but you are willing to sacrifice information on the interactions to obtain maximal information on the main effects. When an orthogonal design of resolution 4 exists, it is optimal according to this Bayesian criterion.

You can use the following statements to generate the Bayesian D-optimal design:

```
proc factex;
   factors x1-x7;
   output out=can;
run;

proc optex data=can seed=57922
           coding=orth;
   model x1-x7,
         x1|x2|x3|x4|x5|x6|x7@@2
         / prior=0,16;
   generate n=20 method=m_fedorov;
   output out=des;
run;
```

With orthogonal coding, the value of the prior for an effect says roughly how many prior "observations' worth" of information you have for that effect. In this case, the PRIOR= precision values and the use of commas to group effects in the MODEL statement says that there is no prior information for the main effects and 16 runs' worth of information for each two-factor interaction. See "Design Coding" on page 774 for details on orthogonal coding.

The efficiencies are shown in Output 24.6.1.

Output 24.6.1. Efficiencies for Bayesian Optimal Designs

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	85.1815	74.6705	85.2579	1.1476
2	85.1815	74.6705	85.2579	1.1476
3	85.1815	74.6705	85.2579	1.1476
4	85.0424	73.3109	81.0800	1.1582
5	85.0424	73.3109	81.0800	1.1582
6	84.5680	73.5053	84.1376	1.1566
7	84.4931	72.1671	81.7855	1.1673
8	84.4239	72.4979	81.7431	1.1646
9	84.3919	74.6097	89.3631	1.1480
10	84.3919	74.6097	89.3631	1.1480

Notice that the best design was found in three tries out of ten. It may be a good idea to repeat the search with more tries (see the ITER= option on page 740.) You can use the ALIASING option of the GLM procedure to list the aliasing structure for the design:

```
data des; set des;
   y = ranuni(654231);
proc glm data=des;
   model
      y = x1-x7
          x1|x2|x3|x4|x5|x6|x7@@2
          / e aliasing;
run;
```

The relevant part of the output is shown in Output 24.6.2. Most of the main effects are indeed unconfounded with two-factor interactions, although many two-factor interactions are confounded with each other.

Output 24.6.2. Aliasing Structure for Bayesian Optimal Design

```
  General Linear Models Procedure
General Form of Aliasing Structure

Intercept
x1 - 0.5*x3*x7
x2
x3
x4 + 0.5*x3*x7
x5
x6
x7
x1*x2 - x3*x6 + 0.5*x3*x7 - x4*x7
x1*x3 - x2*x6 - x5*x7
x2*x3 + x3*x7
x1*x4 - x5*x6 + x5*x7 + x6*x7
x2*x4 - x3*x6 + 0.5*x3*x7 - x4*x7
x3*x4 - x2*x6 - x5*x7
x1*x5 - x4*x6 - x3*x7
x2*x5 + x2*x6 + x5*x7 + x6*x7
x3*x5 + x3*x6 - x3*x7
x4*x5 - x1*x6 - x3*x7
x1*x7 - x4*x7
x2*x7 + x5*x7 + x6*x7
```

Example 24.7. Balanced Incomplete Block Design

See OPTEX8 in the SAS/QC Sample Library

This example uses the BLOCKS statement to construct an incomplete block design. An incomplete block design is a design for v qualitative treatments in b blocks of k runs each, where $k < v$ so that not all treatments can occur in each block. An incomplete block design is said to be *balanced* when all pairs of treatments occur equally often in the same block. A balanced design is always optimal for any criterion based on the information matrix, although there are many values of (v, b, k) for which no balanced design exists.

One way to construct an incomplete block design with the OPTEX procedure is to include the blocking factor in the candidate set and in the model. For example, the following statements search for a BIBD for seven treatments in seven blocks of size three—that is, $(v, b, k) = (7, 7, 3)$—using the full set of 49 treatment-by-block combinations for candidates:

```
data can;
   do tmt = 1 to 7;
      do blk = 1 to 7;
         output;
         end;
      end;
```

```
proc optex data=can seed=8327
           coding=orth;
   class tmt blk;
   model tmt blk;
   generate n=21;
run;
```

By default, the OPTEX procedure performs the search 10 times from different random starting designs. The various efficiencies for each design are listed in Output 24.7.1.

Output 24.7.1. Efficiency Factors for $v = b = 7$, $k = 3$ Designs

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	89.0483	79.1304	82.7170	0.8845
2	89.0483	79.1304	82.7170	0.8845
3	88.4669	76.9882	78.6796	0.8967
4	88.4669	76.9882	78.6796	0.8967
5	88.4669	76.9882	78.6796	0.8967
6	88.4669	76.9882	78.6796	0.8967
7	88.4669	76.9882	78.6796	0.8967
8	88.4669	76.9882	78.6796	0.8967
9	88.1870	76.0262	78.7612	0.9024
10	87.7681	74.2459	73.9544	0.9131

Since the efficiency factors compare the designs to a (hypothetical) orthogonal design, values of 100% are not possible in this case. The OPTEX procedure includes facilities for examining the information matrix for the design; you can use these to verify that the best design found here is, in fact, balanced.

Searching for an optimal design for both treatments and blocks simultaneously has its limitations. Note that the balanced design was found on only two of the ten tries. A more serious limitation is that this approach sometimes fails to find a design with equal-sized blocks. A more efficient and flexible way to construct a block design with the OPTEX procedure is to use the BLOCKS statement.

The following statements use the BLOCKS statement to solve the incomplete block design problem described previously. In this case, the candidate set simply consists of the seven treatment levels.

```
data can;
   do tmt = 1 to 7;
      output;
      end;
proc optex data=can seed=73462
           coding=orth;
   class tmt;
   model tmt;
   blocks structure=(7)3;
run;
```

The output again consists of efficiency factors for 10 different tries, but this time the factors are computed from the information matrix for only the treatment effects. In this special case (a single classification effect in the treatment model together with the BLOCKS STRUCTURE= specification), the efficiency of each design as an incomplete block design is also listed (Output 24.7.2).

Output 24.7.2. Efficiency Factors for $v = b = 7$, $k = 3$ Optimal Blocking Designs

Design Number	Treatment D-Efficiency	Treatment A-Efficiency	Block Design D-Efficiency
1	77.7778	77.7778	100.0000
2	77.7778	77.7778	100.0000
3	77.7778	77.7778	100.0000
4	77.7778	77.7778	100.0000
5	77.7778	77.7778	100.0000
6	77.7778	77.7778	100.0000
7	77.7778	77.7778	100.0000
8	77.7778	77.7778	100.0000
9	77.7778	77.7778	100.0000
10	77.7778	77.7778	100.0000

The 100% efficiency in the fourth column of the output shows that the balanced design was found on all 10 tries.

Since the OPTEX procedure is interactive, you can save the final design in a data set by submitting the OUTPUT statement immediately after the preceding statements. The following statements use the BLOCKNAME= option to rename the block variable:

```
   output out=bibd blockname=blk;
proc print data=bibd;
run;
```

The final design is shown in Output 24.7.3.

Although there is no guarantee that the OPTEX procedure will find the globally optimal block design by this method, it usually does find small- to medium-sized balanced designs, and it always finds a very efficient design. For example, for the designs given in Table 9.5 of Cochran and Cox (1957), the OPTEX procedure consistently finds the theoretically optimal BIBD in all cases with 10 or fewer treatments. Furthermore, in no case is the D-efficiency relative to the balanced design less than 99%.

Output 24.7.3. Balanced Incomplete Block Design for $v = b = 7$, $k = 3$

Obs	BLK	tmt
1	1	1
2	1	4
3	1	7
4	2	6
5	2	3
6	2	1
7	3	2
8	3	5
9	3	1
10	4	6
11	4	2
12	4	7
13	5	5
14	5	4
15	5	6
16	6	5
17	6	7
18	6	3
19	7	4
20	7	3
21	7	2

Example 24.8. Optimal Design with Fixed Covariates

See OPTEX9 in the SAS/QC Sample Library

In addition to finding optimal block designs, you can use the BLOCKS statement to find designs that are optimal with respect to more general covariate models. You can specify the data set containing the covariates with the DESIGN= option in the BLOCKS statement. Covariate models are specified in the same way as the treatment model.

The following example is based on an example in Harville (1974). Suppose you want a design for five qualitative treatments in 10 runs. The value of a covariate thought to be related to the response has been recorded for each of the experimental units. For instance, if the treatments are different types of animal feed, a typical covariate might be the initial weight of each animal. In the following, the data sets COV and TMT are created, containing the covariate values and the candidate treatment levels, respectively. Then the OPTEX procedure is invoked with a simple one-way model for the treatment effect and a quadratic model for the covariate effect.

```
data cov; input u @@@@; datalines;
.46 .54 .58 .60 .73 .77 .82 .84 .89 .95
;
data tmt;
   do t = 1 to 5;
      output;
      end;
proc optex data=tmt seed=17364
           coding=orthcan;
   class t;
   model t;
   blocks design=cov;
   model u u*u;
```

```
   output out=tmtu;
proc print data=tmtu;
run;
```

In this case, the CODING=ORTHCAN option in the PROC OPTEX statement has the same effect as CODING=ORTH, which is to produce orthogonal coding with respect to the candidates. Note that

- the CLASS and MODEL statements that define the treatment model precede the BLOCKS statement
- the MODEL statement that defines the covariate model follows the BLOCKS statement

As a general rule, CLASS and MODEL statements that come before a BLOCKS statement are interpreted as applying to the treatment model, while CLASS and MODEL statements that come after a BLOCKS statement involving the DESIGN= blocks-specification are interpreted as applying to the covariate model.

The listing of the efficiency values for the 10 designs found is shown in Output 24.8.1. Note that the efficiencies are the same for all tries. A listing of the design is shown in Output 24.8.2.

Output 24.8.1. Optimal Treatment Efficiency Factors with a Quadratic Covariate Effect

Design Number	Treatment D-Efficiency	Treatment A-Efficiency
1	91.6621	91.1336
2	91.6621	91.1336
3	91.6621	91.1336
4	91.6621	91.1336
5	91.6621	91.1336
6	91.6621	91.1336
7	91.6621	91.1336
8	91.6621	91.1336
9	91.6621	91.1336
10	91.6621	91.1336

Output 24.8.2. Optimal Design with a Quadratic Covariate Effect

Obs	u	t
1	0.46	4
2	0.54	3
3	0.58	1
4	0.60	2
5	0.73	5
6	0.77	4
7	0.82	3
8	0.84	1
9	0.89	2
10	0.95	5

When you use the BLOCKS statement without specifying the GENERATE statement, the full candidate set is used as the treatment set for optimal blocking. If you specify both statements, an optimal design for the treatments ignoring the blocks is first generated, and the result is used as the treatment set for optimal blocking. This allows several options to be combined to evaluate existing designs. For example, the following statements evaluate the optimal design given in Harville (1974) for the preceding situation:

```
data har; input t @@@@; datalines;
1   2   3   4   5   1   2   3   4   5
;
proc optex data=tmt coding=orthcan;
   class t;
   model t;
   generate initdesign=har
            method=sequential;
   blocks design=cov init=chain iter=0;
   model u u*u;
run;
```

The efficiency values for Harville's design are shown in Output 24.8.3. They are the same as for the design found by the OPTEX procedure.

Output 24.8.3. Treatment Efficiency Factors for Harville's Design

Design Number	Treatment D-Efficiency	Treatment A-Efficiency
1	91.6621	91.1336

In fact, the optimal design found by OPTEX can be derived from Harville's design simply by re-labeling treatments. In order of increasing U, both designs consist of two consecutive replicates of the treatments, with treatments in both replicates occurring in the same order.

Example 24.9. Optimal Design in the Presence of Covariance

See OPTEX10 in the SAS/QC Sample Library

The BLOCKS statement finds a design that maximizes the determinant $|X'AX|$ of the treatment information matrix, where A depends on the block or covariate model. Alternatively, you can directly specify the matrix A to find the D-optimal design when A is the variance-covariance matrix for the runs. You can specify the data set containing the covariance matrix with the COVAR= option in the BLOCKS statement, listing the variables corresponding to the columns of the covariance matrix in the VAR= option. If you specify n variables in the VAR= option, the values of these variables in the first n observations in the data set will be used to define A.

For example, suppose you want to compare the effects of seven different fertilizers on crop yield, using seven long, narrow blocks of four plots each, as depicted in Figure 24.1 on page 761.

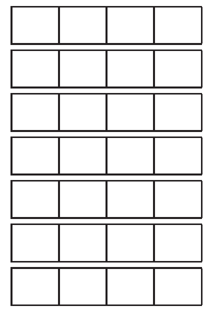

Figure 24.1. Block Structure for Neighbor Balance

In this case, it is reasonable to conjecture that closer plots within each block are more correlated. In particular, suppose that the plots are *autocorrelated*, so that the correlation matrix for the four plots in each block is of the form

$$R = \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

where $-1 \leq \rho \leq 1$. If there is also an overall fixed effect due to blocks, the information matrix for the effect of fertilizer has the form $X'AX$, where

$$A = \left(V^{-1} - V^{-1}Z\left(Z'V^{-1}Z\right)^{-1}Z'V^{-1}\right)^{-}$$

In this formula, V is the block diagonal matrix of the plot-by-plot correlation structure, with seven copies of R_4 on the diagonal. The matrix Z is the design matrix corresponding to the block effect. The optimal design should take into account this neighbor covariance structure as well as the block structure.

The following code uses the SAS/IML matrix language to construct A using $\rho = 0.1$ and saves it in a data set named A:

```
proc iml;
   blks = int(((1:28)‘-1)/4) + 1;
   z = j(28,1) || designf(blks);

   r = toeplitz(0.1**(0:3));
   v = r;
   do i = 2 to 7; v = block(v,r); end;

   iv = inv(v);
   a = ginv(iv-iv*z*inv(z‘*iv*z)*z‘*iv);
   create A from a;
   append from a;
quit;
```

Note that the data set is created with variables named COL1, COL2, . . . , COL28, by default.

To find an allocation of fertilizers to plots that is optimal for detecting the fertilizer effect in the presence of this autocorrelation, simply specify a one-way model for the treatment effects and specify the data set A as the covariance matrix for the runs with the COVAR= option in the BLOCKS statement, as follows:

```
data fert; do f = 1 to 7; output; end;
proc optex data=fert seed=56672 coding=orth;
   class f;
   model f;
   blocks covar=A
          var=(col1-col28);
   output out=nbd;
run;
```

The SAS/IML matrix language also provides a convenient way of listing the design.

```
proc iml;
   use nbd;                                Read in the selected levels
   read all var {f};                          of fertilizer
   nbd = shape(f,7,4);                     Reshape them into 7 4-run
   print nbd [format=2.];                     blocks and print.
```

The resulting design is shown in Output 24.9.1. Note that it is not only a balanced incomplete block design, but it is also balanced for first neighbors; that is, every pair of treatments occur equally often on horizontally adjacent plots.

Output 24.9.1. Neighbor-Balanced BIBD for $v = b = 7$, $k = 4$, Found by Optimal Blocking

```
      NBD

 7  2  1  5
 6  1  7  3
 4  7  6  2
 1  4  6  5
 6  3  5  2
 1  3  2  4
 7  5  4  3
```

Example 24.10. Adding Space-Filling Points to a Design

See OPTEX11 in the SAS/QC Sample Library

Suppose you want a 15-run experiment for three mixture factors X1, X2, and X3; furthermore, suppose that X3 cannot account for any more than 75% of the mixture. You can use the ADXXVERT macro (see page 1896) to construct a list of candidate points for the design and then use the OPTEX procedure to select the design runs optimally for a given model. However, information-based criteria such as D- and A-efficiency tend to push the design to the edges of the candidate space, leaving large portions of the interior relatively uncovered. For this reason, it is often a good idea to augment a D-optimal design with some points chosen according to U-optimality, which seeks to cover the candidate region as well as possible.

The collection of macros for experimental designs described on page 1873 includes several macro programs for working with mixture designs. For example, the following statements invoke the ADXXVERT macro to construct and plot the candidate set:

```
%adxgen
%adxmix
%adxxvert(a,x1 / x2 / x3 0 - .75);
data a; set a;
   w1 = -(x1 + x2);
   w2 =  (x1 - x2);
proc plot data=a;
   plot w1*w2;
run;
```

The arguments to the ADXXVERT macro name the data set to contain the candidate points and list the factors in the design. The form of the list of factors says that X1 and X2 have the default low and high levels of 0 and 1, while the value of X3 is constrained to be between 0 and 0.75. The constraint that the factor levels sum to 1 means that the candidate points all lie on a certain (two-dimensional) plane. The transformed variables W1 and W2 are the coordinates of each candidate point with respect to two orthogonal axes in that plane.

The result, shown in Output 24.10.1, is a “quick-and-dirty” plot of the vertices, the edge centroids, and the over-all centroid for the feasible region. The $X3 \leq 0.75$ constraint effectively “cuts off” the top of the usual simplex.

Output 24.10.1. Vertices and Centroids for Constrained Mixture Design

```
-0.250 +                      A      A      A
       |
       |
       |
       |
    w1 |
       |
       |
       |
       |
       |
-0.625 +            A                A                  A
       |
       |
       |
       |
       |
       |
       |
       |
       |
       |
-1.000 +  A                          A                           A
       ---+-------------+-------------+-------------+-------------+--
        -1.0          -0.5           0.0           0.5           1.0

                                     w2
```

You could easily use this plot to choose 15 runs both to span the extremes of the candidate region and to cover the interior. However, you can use the methods discussed in this section with higher dimensional problems that are difficult or impossible to visualize.

You can use the OPTEX procedure to select 10 optimal points for estimating a second-order model in the mixture factors.

```
proc optex data=a seed=60868 nocode;
   model x1|x2|x3@@2 / noint;
   generate n=10;
   output out=b;
data b; set b;
   w1 = -(x1 + x2);
   w2 =  (x1 - x2);
proc plot data=b;
   plot w1*w2;
run;
```

As shown in Output 24.10.2, the D-optimal design omits some of the candidate points and replicates others.

Output 24.10.2. D-optimal Constrained Mixture Design

```
  -0.250 +                  A           A
         |
         |
         |
         |
     w1  |
         |
         |
         |
         |
         |
  -0.625 +          A                            A
         |
         |
         |
         |
         |
         |
         |
         |
         |
         |
  -1.000 +  B                      B                      B
         ---+-----------+-----------+-----------+-----------+--
          -1.0        -0.5         0.0         0.5         1.0

                                  w2
```

The D-optimal design leaves large “holes” in the feasible region. The following statements use the ADXFILL macro with the candidate set to produce a set of points scattered throughout the feasible region:

```
%adxfill(a,x1 x2 x3);
data a; set a;
   w1 = -(x1 + x2);
   w2 =  (x1 - x2);
proc plot data=a;
   plot w1*w2;
run;
```

The ADXFILL macro simply computes all pairwise averages of points in A and appends them to A; see page 1892 for more details. The results are shown in Output 24.10.3.

Output 24.10.3. Filled Candidate Region for Constrained Mixture Design

```
    w1 |
-0.250 +                       A   A  A   A  A
       |
       |
       |
       |
-0.438 +                  A  A   A A  A   AA   A  A
       |
       |
       |
       |
-0.625 +             A  A   AA     A  A   A    A A  A   A
       |
       |
       |
       |
-0.813 +       A        A    A   A    A    A   A    A        A
       |
       |
       |
       |
-1.000 +  A             A             A             A             A
       |
       ---+-------------+-------------+-------------+-------------+--
        -1.0          -0.5           0.0           0.5           1.0

                                     w2
```

The filled-in data set A has too many points (recall that the goal is a design with 15 runs), but you can use the OPTEX procedure to choose points from it. The following statements "fill in the holes" in the optimal design saved in B by augmenting it with points chosen from the filled-in data set A to optimize the U-criterion:

```
proc optex data=a seed=4321 nocode;
   model x1 x2 x3 / noint;
   generate n=15 augment=b
            criterion=u;
   output out=c;
data c; set c;
   w1 = -(x1 + x2);
   w2 =  (x1 - x2);
proc plot data=c;
   plot w1*w2;
run;
```

Output 24.10.4 shows that the U-optimal design fills in the candidate region in much the same way that you might construct the design by visually assigning points. That is, the general approach using the OPTEX procedure agrees with visual intuition for this small problem. Moreover, the general approach yields an appropriate design for higher dimensional problems that cannot be visualized.

Output 24.10.4. D-optimal Constrained Mixture Design Filled In U-optimally

```
    w1 |
-0.250 +                       A             A
       |
       |
       |
       |
-0.438 +                         A         A
       |
       |
       |
       |
-0.625 +             A             A                     A
       |
       |
       |
       |
-0.813 +                     A                 A
       |
       |
       |
       |
-1.000 +  B                           B                           B
       |
       ---+-------------+-------------+-------------+-------------+--
        -1.0          -0.5           0.0           0.5           1.0

                                     w2
```

Data Details

Input Data Sets

This section discusses the five input data sets for the OPTEX procedure. Three of the data sets provide points used to generate the design according to the effects you specify in the MODEL statement. Two other data sets provide points used to generate a model for fixed covariates.

Only the DATA= data set is required. If you do not specify a DATA= data set in the PROC OPTEX statement, the procedure uses the last data set created as a set of candidate points for the design. The AUGMENT= data set iqs optional and contains points that must be included in the final design. The INITDESIGN= data set is also optional and provides an initial design to be used by a search procedure. Variables listed in the MODEL statement must be present in all three of these data sets, and the variable characteristics (type and length) must match across data sets.

Figure 24.2 is a schematic diagram of the roles of the DATA=, AUGMENT=, and INITDESIGN= data sets in constructing the design. Figure 24.3 presents the role of the DESIGN= data set for block designs.

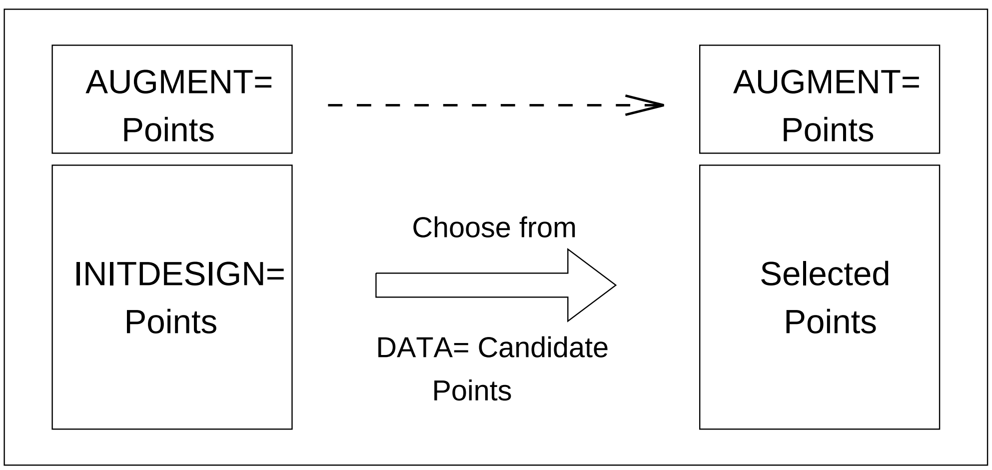

Figure 24.2. Choosing from DATA= Points

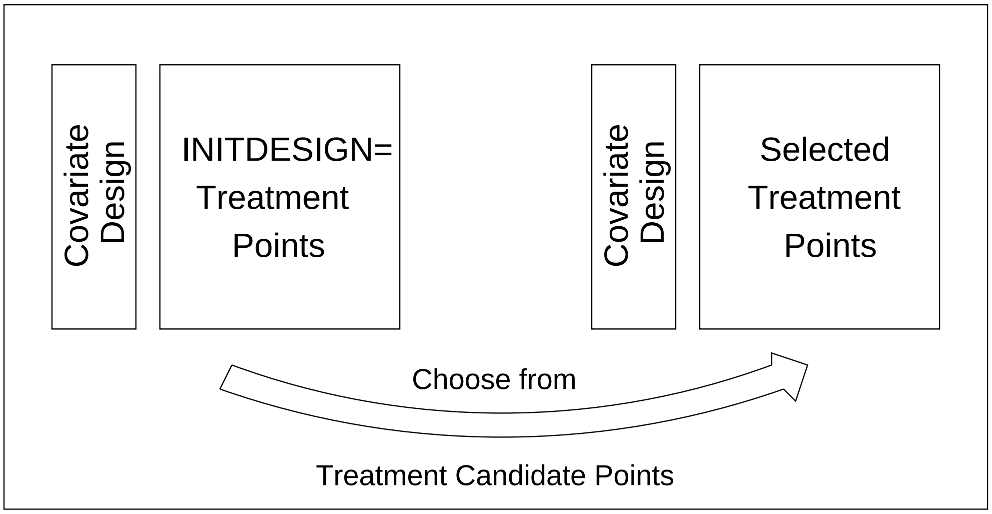

Figure 24.3. Choosing Treatment Candidates

DATA= Data Set

The DATA= data set provides a set of candidate points used to create a design. The OPTEX procedure uses the variables listed in the MODEL statement when creating a design.

The effects specified in a MODEL statement determine the variables used when generating a design. For example, if the DATA= data set contains the variables A, B, and C, but the MODEL statement specifies effects involving only A and B, then the variable C is not considered when generating designs.

Variables in the DATA= data set that are listed in the ID statement are transferred to the OUT= data set (if one is created).

AUGMENT= Data Set

The AUGMENT= data set provides a set of points that must be included in the final design. The OPTEX procedure adds candidate points from the DATA= data set to the points from the AUGMENT= data set when generating designs. The number of points in the AUGMENT= data set must be less than or equal to the number of points for the design (either the default or the number specified with the N= option in the GENERATE statement).

As with the DATA= data set, the effects specified in a MODEL statement determine the variables used when generating a design. The types and lengths of variables in an AUGMENT= data set that are used in the MODEL and ID statements must match the types and lengths of the same variables in the DATA= data set. If you use an ID statement and the AUGMENT= data set contains the ID variables, these variables are transferred to the OUT= data set (if one is created). See “Including Specific Runs” on page 724 for an example that uses an AUGMENT= data set.

INITDESIGN= Data Set

The INITDESIGN= data set provides a set of points that are used as an initial design in the search for an optimal design. These points are not necessarily contained in the final design. The OPTEX procedure uses these points to begin the search for an optimal design. The number of points in the INITDESIGN= data set must be the same as the number of points in the design (either the default or the number specified with the N= option in the GENERATE statement).

As with the DATA= data set, the effects specified in a MODEL statement determine the variables used when generating a design. The types and lengths of variables in an INITDESIGN= data set that are used in the MODEL and ID statements must match the types and lengths of the same variables in the DATA= data set. If you use an ID statement and the INITDESIGN= data set contains the ID variables, these variables are transferred to the OUT= data set (if one is created). See Example 24.3 on page 749 for an example that uses an INITDESIGN= data set.

If you use an INITDESIGN= data set and also specify METHOD=SEQUENTIAL in the GENERATE statement, no search is performed. The INITDESIGN= data set is the final design. In this way, you can use the OPTEX procedure to evaluate an existing design.

BLOCKS DESIGN= Data Set

The DESIGN= data set in the BLOCKS statement contains a set of points that are used to generate a model for fixed covariates. These points are contained in the final design and are transferred to the OUT= data set (if one is created). See Example 24.8 on page 758 for an example that uses a BLOCKS DESIGN= data set.

BLOCKS COVAR= Data Set

If you specify a COVAR= data set in the BLOCKS statement, the observations for the variables listed in the VAR= option are used to define the assumed variance-covariance matrix for the experimental runs. These observations are *not* transferred to the OUT= data set (if one is created). Note that since covariance matrices are necessarily square, the number of observations in the COVAR= data set must be the same as the number of variables listed in the VAR= option. See Example 24.9 on page 760 for an example that uses a BLOCKS COVAR= data set.

Output Data Sets

You typically use the OPTEX procedure to create an output data set that contains the design for your experiment. If you use an OUTPUT statement, the variables in the output data set are the factors of the design as well as any ID variables. The values for the ID variables are taken from the input data set (the DATA=, AUGMENT=, or INITDESIGN= data set) that provided the design point. ID variables must be contained in the DATA= data set and can also be contained in the AUGMENT= or INITDESIGN= data sets. If an AUGMENT= or INITDESIGN= data set does not contain the ID variables, and points from the data set are used in the final design, values of ID variables for those points are missing.

Since the input data sets provide candidate points for the design, all the observations in the OUT= data set originate in one of the input data sets. The OPTEX procedure does not change the values of variables in the input data sets.

Since you can use multiple OUTPUT statements with the OPTEX procedure, you can create multiple OUT= data sets in a given run of the procedure.

Computational Details

Specifying Effects in MODEL Statements

This section discusses how to specify the linear model that you plan to fit with the design. The OPTEX procedure provides for the same general linear models as the GLM procedure, although it does not use the GLM procedure's *over-parameterized* technique for generating the design matrix (see "Static Coding" on page 774.)

Each term in a model, called an *effect*, is a variable or combination of variables. To specify effects, you use a special notation involving variables and operators. There are two kinds of variables: *classification variables* and *continuous variables*. *Classification variables* separate observations into groups, and the model depends on them through these groups; on the other hand, the model depends on the actual (or coded) values of *continuous variables*. There are two primary operators: *crossing* and *nesting*. A third operator, the *bar operator*, simplifies the specification for multiple crossed terms, as in a factorial model. The `@` operator, used in combination with the bar operator, further simplifies specification of crossed terms.

When specifying a model, you must list the classification variables in a CLASS statement. Any variables in the model that are not listed in the CLASS statement are assumed to be continuous. Continuous variables must be numeric.

Types of Effects

Five types of effects can be specified in the MODEL statement. Each row of the design matrix is generated by combining values for the independent variables according to effects specified in the MODEL statement. This section discusses how to specify different types of effects and explains how they relate to the columns of the design matrix. In the following, assume that A, B, and C are classification variables and X1, X2, and X3 are continuous variables.

Regressor Effects

Regressor effects are specified by writing continuous variables by themselves.

```
X1  X2  X3
```

For regressor effects, the actual values of the variable are used in the design matrix.

Polynomial Effects

Polynomial effects are specified by joining two or more continuous variables with asterisks.

```
X1*X1  X1*X1*X1  X1*X2  X1*X2*X3  X1*X1*X2
```

Polynomial effects are also referred to as interactions or cross products of continuous variables; when a variable is joined with itself, polynomial effects are referred to as quadratic effects, cubic effects, and so on. In the preceding examples, the first two effects are the quadratic and cubic effects for X1, respectively. The remaining effects are cross products.

For polynomial effects, the value used in the design matrix is the product of the values of the constituent variables.

Main Effects

If a classification variable A has k levels, then its main effect has $k-1$ degrees of freedom, corresponding to $k-1$ independent differences between the mean response at different levels. Main effects are specified by writing class variables by themselves.

```
A  B  C
```

Most designs involve main effects since these correspond to the factors in your experiment. For example, in a factorial design for a chemical process, the main effects may be temperature, pressure, and the level of a catalyst.

For information on how the OPTEX procedure generates the $k-1$ columns in the design matrix corresponding to the main effect of a classification variable, see “Design Coding” on page 774.

Crossed Effects

Crossed effects (or interactions) are specified by joining class variables with asterisks.

```
A*B  B*C  A*B*C
```

The number of degrees of freedom for a crossed effect is the product of the numbers of degrees of freedom for the constituent main effects. The columns in the design matrix corresponding to a crossed effect are formed by the horizontal direct products of the constituent main effects.

Continuous-by-Class Effects

Continuous-by-class effects are specified by joining continuous variables and class variables with asterisks.

```
X1*A
```

The design columns for a continuous-by-class effect are constructed by multiplying the values in the design columns for the continuous variables and the class variable.

Note that all design matrices start with a column of ones for the assumed intercept term unless you use the NOINT option in the MODEL statement.

Bar and @ Operators

You can shorten the specification of a factorial model using the bar operator. For example, the following statements show two ways of specifying a full three-way factorial model:

```
model a b c a*b a*c b*c a*b*c;
model a|b|c;
```

When the vertical bar (**|**) is used, the right- and left-hand sides become effects, and their cross becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right using rules given by Searle (1971). For example, **A|B|C** is evaluated as follows:

$$
\begin{aligned}
\texttt{A|B|C} &\rightarrow \{\texttt{A|B}\}\texttt{|C} \\
&\rightarrow \{\texttt{A B A*B}\}\texttt{|C} \\
&\rightarrow \texttt{A B A*B C A*C B*C A*B*C}
\end{aligned}
$$

The bar operator does not cross a variable with itself. To produce a quadratic term, you must specify it directly.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by putting it at the end of a bar effect, preceded by an **@** sign. For example, the specification **A|B|C@2** results in only those effects that contain two or fewer variables (in this case A, B, A*B, C, A*C, and B*C.)

Examples of Models

Main Effects Model

For a three-factor main effects model with A, B, and C as the factors, the MODEL statement is

```
model a b c;
```

Factorial Model with Interactions

To specify interactions in a factorial model, join effects with asterisks, as described previously. For example, the following statements show two ways of specifying a complete factorial model, which includes all the interactions:

```
model a b c a*b a*c b*c a*b*c;
model a|b|c;
```

Quadratic Model

The following statements show two ways of specifying a model with crossed and quadratic effects (for a central composite design, for example):

```
model x1 x2 x1*x2 x3 x1*x3 x2*x3
      x1*x1 x2*x2 x3*x3;
model x1|x2|x3@@2 x1*x1 x2*x2 x3*x3;
```

Design Efficiency Measures

The output from the OPTEX procedure includes efficiency measures for the resulting designs according to various criteria. This section gives the precise definitions for these measures.

By default, the OPTEX procedure calculates the following efficiency measures for each design found in its search for an optimum design:

$$\begin{aligned}
\text{D-efficiency} &= 100 \times \left(\frac{|X'X|^{1/p}}{N_D} \right) \\
\text{A-efficiency} &= 100 \times \left(\frac{p/N_D}{\text{trace}(X'X)^{-1}} \right) \\
\text{G-efficiency} &= 100 \times \left(\sqrt{\frac{p/N_D}{\max_{\mathbf{x}\in\mathcal{C}} \mathbf{x}'(X'X)^{-1}\mathbf{x}}} \right)
\end{aligned}$$

where p is the number of parameters in the linear model, N_D is the number of design points, and $\mathcal{C}$ is the set of candidate points. The D- and A-efficiencies are the relative number of runs (expressed as percents) required by a hypothetical orthogonal design to achieve the same $|X'X|$ and $\text{trace}(X'X)^{-1}$, respectively; refer to Mitchell (1974b).

When you specify a BLOCKS statement, the D- and A-efficiencies for the treatment part of the model are calculated. These are calculated similarly to the preceding efficiencies, except that they are based on the information matrix after correcting for covariate effects. This matrix can be written as $X'AX$ for a symmetric, positive definite matrix A that depends on the model for the covariate effect. If you specify a block structure or a covariate model, then $A = I - Z(Z'Z)^{-1}Z'$, where Z is the design matrix for the block or covariate effect. Alternatively, you can use the COVAR= option to specify the matrix A directly. Given A, the efficiencies in the presence of covariates are defined as follows:

$$\begin{array}{lcllcl}
\text{D-efficiency} & = & 100 \times c_D^{-1} \cdot |X'AX|^{1/p}/N, & c_D & = & \prod_{i=1}^{p} \lambda_i^{1/p} \\
\text{A-efficiency} & = & 100 \times c_A \cdot (p/N)/\text{trace}(X'AX)^{-1}, & c_A & = & \sum_{i=1}^{p} \lambda_i/p
\end{array}$$

where $\lambda_1, \ldots, \lambda_p$ are the p largest eigenvalues of A. If you use the STRUCTURE= block model specification and there is only one class variable in the treatment model, then the design fits into the traditional block design framework. In this case, the D-efficiency relative to a balanced incomplete block design is also listed.

Because these efficiencies measure the goodness of the design relative to theoretical designs that may be far from possible in many cases, they are typically not useful as absolute measures of design goodness. Instead, efficiency measures should be used relatively, to compare one design to another for the same situation.

For the distance-based criteria, there are no simple measures of design efficiency that can be scaled from 0 to 100. See the “Output” section on page 783 for a definition of the design measures tabulated for these criteria.

Design Coding

The way the independent effects of the model are interpreted to generate a linear model is called *coding*. The OPTEX procedure provides for different types of coding. For D-optimality, the type of coding affects only the absolute value of the computed efficiency criteria, not the relative values for two different designs. Thus, different codings do not affect the choice of D-optimal design. In this section, the details and ramifications of the different types of coding are discussed.

Coding the points in a design involves selecting linearly independent columns corresponding to each model term, turning particular values of the factors into a row vector $\mathbf{x}$. The OPTEX procedure requires a *non-singular* coding for the design matrix. Because of this, any two coding schemes are related by a non-singular transformation.

Static Coding

The default coding for the design points is as follows:

- Unless you specify CODING=NONE (or NOCODE) in the PROC OPTEX statement, continuous variables are centered and scaled so that their maximum and minimum values are 1 and –1, respectively.
- The $k-1$ columns corresponding to the main effect of a classification variable A are computed as follows: For a design point with A at its i^{th} level, for $1 \leq i \leq k-1$, the columns of the design matrix associated with A are all 0 except for the i^{th} column, which is 1. When A is at its k^{th} level, all $k-1$ columns associated with A are -1. Thus, if α_i denotes the expected response at the i^{th} level of A, the $k-1$ columns yield estimates of $\alpha_1-\alpha_k, \alpha_2-\alpha_k, \ldots, \alpha_{k-1}-\alpha_k$.
- Columns for crossed effects are computed by taking the horizontal direct product of columns corresponding to the constituent effects.

This coding corresponds to modeling without *over-parameterization*, using the same method as the CATMOD procedure in SAS/STAT software. This is different from the method used by the GLM procedure, which uses an over-parameterized model.

Orthogonal Coding

If you specify CODING=ORTH or CODING=ORTHCAN, the points are first coded as described in the previous section and then recoded so that $X_C'X_C = N_C \cdot I$, where X_C is the design matrix for the candidate points, N_C is the number of candidates, and I is the identity matrix. This is required in order for the D- and A-efficiency measures to make sense. For the option CODING=ORTHCAN, this recoding is accomplished by computing a square matrix R such that $X_C'X_C = R'R$ and then transforming each row vector $\mathbf{x}$ as

$$\mathbf{x} \quad \rightarrow \quad \mathbf{x}R^{-1}\sqrt{N_C}$$

If you specify CODING=ORTH, the recoding is done in a similar fashion, except that the matrix R is computed according to $X_C'X_C + X_A'X_A + X_I'X_I = R'R$, where X_A and X_I are the design matrices (coded as described in the previous section.) Thus, these two orthogonal coding options only differ when there is an AUGMENT= or INITDESIGN= data set (see pages 738–739); the option CODING=ORTH includes points from these data sets in computing the orthogonal coding, while the option CODING=ORTHCAN uses only the candidates themselves.

Example of Coding

For example, consider a main effect model with one continuous variable X and one three-level classification variable A. The results of the various coding options are shown in Figure 24.4.

Original Data	
X	**A**
1	1
2	2
3	3
4	1
5	2
6	3

Design Matrix With CODING=NONE			
	X	**A1**	**A2**
1	1	1	0
1	2	0	1
1	3	–1	–1
1	4	1	0
1	5	0	1
1	6	–1	–1

Design Matrix With CODING=STATIC			
	X	**A1**	**A2**
1	–1	1	0
1	–0.6	0	1
1	–0.2	–1	–1
1	0.2	1	0
1	0.6	0	1
1	1	–1	–1

Design Matrix With CODING=ORTH			
	X	**A1**	**A2**
1	–1.464	0.598	–0.707
1	–0.878	–0.478	1.414
1	–0.293	–1.554	–0.707
1	0.293	1.554	–0.707
1	0.878	0.478	1.414
1	1.464	–0.598	–0.707

Figure 24.4. Different Types of Design Coding

The first column in each design matrix is an all-ones vector corresponding to the intercept, the next column corresponds to the linear effect of X, and the last two columns correspond to the two degrees of freedom for the main effect of A.

General Recommendations

Coding does not affect the relative ordering of designs by D-efficiency, and the same is true for G-efficiency and the average standard error of prediction. This is easy to see for the latter two measures, which are based on the variance of prediction, since how accurately a point is predicted should not be affected by how the independent variables are coded. For D-optimality, note again that coding corresponds to multiplying the design matrix on the right by some non-singular transformation A, which changes the determinant of the information matrix as follows:

$$|X'X| \quad \rightarrow \quad |A'X'XA| \;=\; |A'A||X'X| \;=\; |A|^2|X'X|$$

Thus, recoding simply multiplies the D-criterion by a constant that is the same for all designs. Note, however, that A-optimality is *not* invariant to coding.

Orthogonal coding will usually be the right one; it is not the default because it depends on the candidate set. Note, however, that for the distance-based criteria, if the distance between two points should be computed in terms of the actual values of the model variables instead of centered and scaled values, then you should specify CODING=NONE or NOCODE. The NOCODE option is also usually appropriate when the NOINT option is specified.

Optimality Criteria

An optimality criterion is a single number that summarizes how good a design is, and it is maximized or minimized by an optimal design. This section discusses in detail the optimality criteria available in the OPTEX procedure.

Types of Criteria

Two general types of criteria are available: *information-based* criteria and *distance-based* criteria.

The information-based criteria that are directly available are D- and A-optimality; they are both related to the information matrix $X'X$ for the design. This matrix is important because it is proportional to the inverse of the variance-covariance matrix for the least-squares estimates of the linear parameters of the model. Roughly, a good design should "minimize" the variance $(X'X)^{-1}$, which is the same as "maximizing" the information $X'X$. D- and A-efficiency are different ways of saying how large $(X'X)$ or $(X'X)^{-1}$ are.

For the distance-based criteria, the candidates are viewed as comprising a point cloud in p-dimensional Euclidean space, where p is the number of terms in the model. The goal is to choose a subset of this cloud that "covers" the whole cloud as uniformly as possible (in the case of U-optimality) or that is as broadly "spread" as possible (in the case of S-optimality). These ideas of coverage and spread are defined in detail on page 778. The distance-based criteria thus correspond to the intuitive idea of filling the candidate space as well as possible.

The rest of this section discusses different optimality criterion in detail.

D-optimality

D-optimality is based on the determinant of the information matrix for the design, which is the same as the reciprocal of the determinant of the variance-covariance matrix for the least-squares estimates of the linear parameters of the model.

$$|X'X| \quad = \quad 1/|(X'X)^{-1}|$$

The determinant is thus a general measure of the size of $(X'X)^{-1}$. D-optimality is the most common criterion for computer-generated optimal designs, which is why it is the default criterion for the OPTEX procedure.

The D-optimality criterion has the following characteristics:

- D-optimality is the most computationally efficient criterion to optimize for the low-rank update algorithms of the OPTEX procedure, since each update depends only on the variance of prediction for the current design; see "Useful Matrix Formulas" on page 780.
- $|X'X|$ is inversely proportional to the size of a $100(1-\alpha)\%$ confidence ellipsoid for the least-squares estimates of the linear parameters of the model.
- $|X'X|^{1/p}$ is equal to the geometric mean of the eigenvalues of $X'X$.
- The D-optimal design is invariant to non-singular recoding of the design matrix.

$$|X'X| \;\; \rightarrow \;\; |A'X'XA| \; = \; |A'A||X'X| \; = \; |A|^2|X'X|$$

A-optimality

A-optimality is based on the sum of the variances of the estimated parameters for the model, which is the same as the sum of the diagonal elements, or trace, of $(X'X)^{-1}$. Like the determinant, the A-optimality criterion is a general measure of the size of $(X'X)^{-1}$. A-optimality is less commonly used than D-optimality as a criterion for computer optimal design. This is partly because it is more computationally difficult to update; see "Useful Matrix Formulas" on page 780. Also, A-optimality is *not* invariant to non-singular recoding of the design matrix; different designs will be optimal with different codings.

G- and I-optimality

Both G-efficiency and the average prediction variance are well-known criteria for optimal design. Both are based on the variance of prediction of the candidate points, which is proportional to $\mathbf{x}'(X'X)^{-1}\mathbf{x}$. As this formula shows, these two criteria are also related to the information matrix $X'X$. Minimizing the average prediction variance has also been called *I-optimality*, the "I" denoting integration over the candidate space.

It is possible to apply the search techniques available in the OPTEX procedure to these two criteria, but this turns out to be a poor way to find G- and I-optimal designs. One reason for this is that there are no efficient low-rank update rules (see "Useful Matrix Formulas" on page 780), so that the searches can take a very long time. More seriously, for G-optimality such a search often does not converge on a design with good G-efficiency. G-efficiency is simply too "rough" a criterion to be optimized by the relatively short steps of the search algorithms available in the OPTEX procedure.

However, the OPTEX procedure does offer an approach for finding G-efficient designs. Begin by searching for designs according to the default D-optimality criterion. Then, from the various designs found on the different tries, you can save the one that has the best G-efficiency by specifying the NUMBER=GBEST option in the OUTPUT statement. Since D- and G-efficiency are highly correlated over the space of all designs, this method usually results in adequately G-efficient designs, especially when the number of tries is large. See the ITER= option on page 740 for details on specifying the number of tries.

To find I-optimal designs, note that if the design is orthogonally coded then I-optimality is equivalent to the A-optimality, since the sum of the prediction variances of all points $\mathbf{x}$ in the candidate space $\mathcal{C}$ is

$$\begin{aligned}
\sum_{\mathbf{x}\in\mathcal{C}} \mathbf{x}'(X'X)^{-1}\mathbf{x} &= \sum_{\mathbf{x}\in\mathcal{C}} \mathrm{trace}\left(\mathbf{x}'(X'X)^{-1}\mathbf{x}\right) \\
&= \mathrm{trace}\left((X'X)^{-1}\sum_{\mathbf{x}\in\mathcal{C}} \mathbf{x}\mathbf{x}'\right) \\
&= \mathrm{trace}\left((X'X)^{-1}X_C'X_C\right) \\
&= N_C \cdot \mathrm{trace}\left((X'X)^{-1}\right)
\end{aligned}$$

where N_C is the number of candidate points and X_C is the design matrix for the candidate points. Thus, you can use the option CODING=ORTH in the PROC OPTEX statement together with the option CRITERION=A in the GENERATE statement to search for I-optimal designs.

Note that both G- and I-optimality are invariant to non-singular recoding of the design matrix, since the coding does not affect how well a point is predicted.

Distance-based Criteria

The distance-based criteria are based on the distance $d(\mathbf{x},\mathcal{A})$ from a point $\mathbf{x}$ in the p-dimensional Euclidean space $\mathcal{R}^p$ to a set $\mathcal{A} \subset \mathcal{R}^p$. This distance is defined as follows:

$$d(\mathbf{x},\mathcal{A}) = \min_{\mathbf{y}\in\mathcal{A}} ||\mathbf{x}-\mathbf{y}||$$

where $||\mathbf{x}-\mathbf{y}||$ is the usual p-dimensional Euclidean distance,

$$||\mathbf{x}-\mathbf{y}|| = \sqrt{(x_1-y_1)^2+\ldots+(x_p-y_p)^2}$$

U-optimality seeks to minimize the sum of the distances from each candidate point to the design.

$$\sum_{\mathbf{x}\in\mathcal{C}} d(\mathbf{x},\mathcal{D})$$

where $\mathcal{C}$ is the set of candidate points and $\mathcal{D}$ is the set of design points. You can visualize the U criterion by associating with any design point those candidates to which it is closest. Thus, the design defines a *clustering* of the candidate set, and indeed cluster analysis has been used in this context. Johnson, Moore, and Ylvisaker (1990) consider a similar measure of design efficiency, but over infinite rather than finite candidate spaces. Computationally, the U-optimality criterion can be *very* difficult to optimize, especially if the matrix of all pairwise distances between candidate points does not fit in memory. In this case, the OPTEX procedure recomputes each distance as needed. When searching for a U-optimal design, you should start with a small version of the problem to get an idea of the computing resources required.

S-optimality seeks to maximize the harmonic mean distance from each design point to all the other points in the design.

$$\frac{N_D}{\sum_{\mathbf{y}\in\mathcal{D}} 1/d(\mathbf{y},\mathcal{D}-\mathbf{y})}$$

For an S-optimal design, the distances $d(\mathbf{y},\mathcal{D}-\mathbf{y})$ are large, so the points are as spread out as possible. Since the S-optimality criterion depends only on the distances between design points, it is usually computationally easier to compute and optimize than the U-optimality criterion, which depends on the distances between all pairs of candidate points.

Memory and Run-Time Considerations

The OPTEX procedure provides a computationally intensive approach to designing an experiment, and therefore some finesse is called for to make the most efficient use of computer resources.

The OPTEX procedure must retain the entire set of candidate points in memory. This is necessary because all of the search algorithms access these points repeatedly. If this requires more memory than is available, consider using knowledge of the problem to reduce the set of candidate points. For example, for first- or second-order models, it is usually adequate to restrict the candidates to just the center and the edges of the experimental region or perhaps an even smaller set; see the introductory examples on page 727 and page 728.

The distance-based criteria (CRITERION=U and CRITERION=S) also require repeated access to the distance between candidate points. The procedure will try to fit the matrix of these distances in memory; if it cannot, it will recompute them as needed, but this will cause the search to be dramatically slower.

The run time of each search algorithm depends primarily on N_D, the size of the target design and on N_C, the number of candidate points. For a given model, the run times of the sequential, exchange, and DETMAX algorithms are all roughly proportional to both N_D and N_C (that is, $O(N_D)+O(N_C)$). The run times for the two simultaneous switching algorithms (FEDOROV and M_FEDOROV) are roughly proportional to the product of N_D and N_C (that is, $O(N_C N_D)$). The constant of proportionality is larger when searching for A-optimal designs because the update formulas are more complicated (see “Search Methods,” which follows).

For problems where either N_D or N_C is large, it is a good idea to make a few test runs with a faster algorithm and a small number of tries before attempting to use one of the slower and more reliable search algorithms. For most problems, the efficiency of a design found by a faster algorithm will be within one or two percent of that for the best possible design, and this is usually sufficient if it appears that searching with a slower algorithm is infeasible.

Search Methods

The search procedures available in the OPTEX procedure offer various compromises between speed and reliability in finding the optimum. In general, the longer an algorithm takes to arrive at an answer, the more efficient is the resulting design, although this is not invariably true. The right search procedure for any specific case depends on the size of the problem, the relative importance of using the best possible design as opposed to a very good one, and the computing resources available.

Useful Matrix Formulas

All of the search algorithms are based on adding candidate points to the growing design and deleting them from a design that is too big. If $V = (X'X)^{-1}$ is the inverse of the information matrix for the design at any stage, then the change in V that results from adding a new point to the design (which adds a new row $\mathbf{x}$ to the design matrix) is

$$V \quad \rightarrow \quad V - \frac{V\mathbf{x}\mathbf{x}'V}{1+\mathbf{x}'V\mathbf{x}}$$

and the change in V that results from deleting the point $\mathbf{y}$ from the design is

$$V \quad \rightarrow \quad V + \frac{V\mathbf{y}\mathbf{y}'V}{1-\mathbf{y}'V\mathbf{y}}$$

It follows, for example, that adding $\mathbf{x}$ multiplies the determinant of the information matrix by $1+\mathbf{x}'V\mathbf{x}$, and likewise deleting $\mathbf{y}$ multiplies the determinant by $1-\mathbf{y}'V\mathbf{y}$. For any point $\mathbf{z}$, the quantity $\mathbf{z}'V\mathbf{z}$ is proportional to the prediction variance at the point $\mathbf{z}$. Thus, the point $\mathbf{x}$ whose addition to the design maximizes the determinant of the information is the point whose prediction variance calculated from the present design is largest. The point whose deletion from the design costs the least in terms of the determinant is the point with the smallest prediction variance.

Similar rank-one update formulas can be derived for A-optimality, which is based on the trace of the inverse of the information matrix instead of its determinant. However, in this case there is no single quantity that can be examined for both adding and deleting a point. Instead, the trace that results from adding a point $\mathbf{x}$ depends on

$$\frac{\mathbf{x}'V^2\mathbf{x}}{1+\mathbf{x}'V\mathbf{x}}$$

and the trace that results from deleting a point $\mathbf{y}$ depends on

$$\frac{\mathbf{y}'V^2\mathbf{y}}{1-\mathbf{y}'V\mathbf{y}}$$

This complication makes A-optimal designs harder to search for than D-optimal ones.

There are no useful rank-one update formulas for the distance-based design criteria.

Sequential Search Algorithm

The simplest and fastest algorithm is the sequential search due to Dykstra (1971), which starts with an empty design and adds successive candidate points so that the chosen criterion is optimized at each step. You can use the sequential procedure as a first step in finding a design

- to judge the size of the problem in terms of time and space requirements
- to determine the number of design points needed to estimate the parameters of the model

The sequential algorithm requires no initial design; in fact, it can be used to provide an initial design for the other search procedures (see the INITDESIGN= option on page 739). If you specify a data set for an initial design for this search procedure, no search will be made; in this way, the OPTEX procedure can be used to evaluate an existing design.

Since the sequential search method involves no randomness, it requires only one try to find a design. The sequential procedure is by far the fastest of any of the search methods, but in difficult design situations it is also the least reliable in finding a globally optimal design. Also, the fact that it always finds the same design (due to the lack of randomness mentioned previously) makes it inappropriate when you want to find a design that represents a compromise between several optimality criteria.

Exchange Algorithm

The next fastest algorithm is the simple exchange method of Mitchell and Miller (1970). This technique tries to improve an initial design by adding a candidate point and then deleting one of the design points, stopping when the chosen criterion ceases to improve. This method is relatively fast (though typically much slower than the sequential search) and fairly reliable. METHOD=EXCHANGE is the default.

Johnson and Nachtsheim (1983) introduce a generalization of both the simple exchange algorithm and the modified Fedorov search algorithm of Cook and Nachtsheim (1980), which is described later in this list. In the modified Fedorov algorithm, each of the points in the current design is considered for exchange with a candidate point; in the generalized version, only the k design points with smallest variance in the current design are considered for exchange. You can specify k-exchange as the search procedure for OPTEX by giving a value for k in parentheses after METHOD=EXCHANGE. When $k = N_D$, the size of the design, k-exchange is equivalent to the modified Fedorov algorithm; when $k = 1$, it is equivalent to the simple exchange algorithm. Cook and Nachtsheim (1980) indicate that $k < N_D/4$ is typically sufficient.

DETMAX Algorithm

The DETMAX algorithm of Mitchell (1974a) is the best known and most widely used optimal design search algorithm. It generalizes the simple exchange method. Instead of requiring that each addition of a point be followed directly by a deletion, the algorithm provides for *excursions* in which the size of the design may vary between $N_D + k$ and $N_D - k$. Here $N_D + k$ is the specified size of the design and k is the maximum allowed size for an excursion. By default k is 4, but you can change this (see the METHOD=DETMAX(*level*) option on page 740). For the precise stopping rules for each excursion and for the entire search, refer to Mitchell (1974a).

Fedorov and Modified Fedorov Algorithms

The three algorithms discussed so far add and delete points one at a time. By contrast, the Fedorov and modified Fedorov algorithms are based on simultaneous switching, adding and deleting points simultaneously. These two algorithms usually find a better design than the others, but because each step involves a search over all possible pairs of candidate and design points, they generally run much slower.

At each step, the Fedorov algorithm (Fedorov 1972) seeks the pair $(\mathbf{x}, \mathbf{y})$ of one candidate point and one design point that optimizes the change $\Delta(\mathbf{x}, \mathbf{y})$ in the optimality criterion, and then switches $\mathbf{x}$ for $\mathbf{y}$ in the design. Thus, after computing $\Delta(\mathbf{x}, \mathbf{y})$ for all possible pairs of candidate and design points, the Fedorov algorithm performs only one switch.

The modified Fedorov algorithm of Cook and Nachtsheim (1980) computes the same number of Δ's on each step but switches each point $\mathbf{y}$ in the design with the candidate point $\mathbf{x}$ that maximizes Δ**(x,y)**. This procedure is generally as reliable as the simple Fedorov algorithm in finding the optimal design, but it can be up to twice as fast.

Optimal Blocking

Building on the work of Harville (1974), Cook and Nachtsheim (1989) give an algorithm for finding D-optimal designs in the presence of fixed block effects. In this case, the design for the original candidate points is called the *treatment design*; the information matrix for the treatment design has the form $X'AX$ for a certain symmetric, nonnegative-definite matrix A that depends on the blocks. The algorithm is based on two kinds of low-rank changes to the treatment design matrix X: *exchanging* a point in the design with a potential treatment point, and *interchanging* two points in the design. Cook and Nachtsheim (1989) give formulas for computing the resulting change in $X'AX$ and $|X'AX|$. These update formulas can be generalized to apply whenever the information matrix for the treatment design has the form $X'AX$, not just when A is derived from fixed blocks. This is the basis for the optimal blocking algorithm in the OPTEX procedure.

Notice that you can combine several options to use the OPTEX procedure to *evaluate* a design with respect to the fixed covariates. Assume the design you want to evaluate is in a data set named EDESIGN. Then first specify

```
generate initdesign=edesign method=sequential;
```

This makes the data set EDESIGN the treatment design. Then specify the following BLOCKS statement options:

```
blocks {block-specification} init=chain iter=0;
```

The INIT=CHAIN option ensures that the starting ordering for the treatment points is the same as in the EDESIGN data set, and the ITER=0 specification causes the procedure simply to output the efficiencies for the initial design, without trying to optimize it.

Search Strategies

General Recommendations

As with all combinatorial optimization problems, finding efficient experimental designs can be difficult. For this reason, the OPTEX procedure provides a variety of ways to customize the search.

Although default settings make the procedure simple to use "as is," you can usually improve the search using knowledge of the specific design problem. For example, if the default algorithm (EXCHANGE) runs quickly but it is not clear whether it finds the best design, you can try a slower but more reliable search method or use more iterations than the default number of 10.

Set of Candidate Points

The choice of candidate points can profoundly affect both the speed with which the search converges at a local optimum and the likelihood that this local optimum is indeed the global optimum. Up to a point, the more candidate points there are, the better the resulting optimum design will be but the longer it will take to find. Any prior knowledge that can be brought to bear on the choice of candidates will almost certainly improve the search. For example, for first- or second-order models it is usually adequate to restrict the candidates to just the center and the edges of the experimental region, or perhaps even less; refer to Snee (1985), and see the introductory examples on page 727 and page 728.

Initial Design

The reliability of the search algorithms in finding the optimal design can be quite sensitive to the choice of initial design. The default method of initialization for each search procedure should achieve good results for a wide variety of situations (see the INITDESIGN= option on page 739). However, in certain situations it is better to override the defaults. For example, if there are many local optima and you want to find the exact global optimum, it will probably be best to start each try with a completely random design (INITDESIGN=RANDOM). On the other hand, prior knowledge may provide a specific initial design, which can be placed in a SAS data set and specified with the INITDESIGN= option.

Output

By default, the OPTEX procedure lists the following information for each attempt to find the optimum design:

- the D-efficiency of the design
- the A-efficiency of the design
- the G-efficiency of the design
- the square root of the average variance for prediction over the candidate points

If you specify a BLOCKS statement, then the covariate-adjusted D- and A-efficiencies are also listed.

See "Design Efficiency Measures" on page 773 for details on the efficiencies. The OPTEX procedure orders the designs first by the optimality criteria with which they were generated and then by optimality with respect to the other three preceding measures.

If you use the NOCODE option, the OPTEX procedure lists

- $\log |\mathbf{X}'\mathbf{X}|$
- $\mathrm{trace}(X'X)^{-1}$
- the G-efficiency of the design
- the square root of the average variance for prediction over the candidate points

If you specify one of the distance-based optimality criteria (CRITERION=U or CRITERION=S), then, instead of the preceding efficiencies, alternative measures of coverage and spread are listed. For U-optimality these measures are

- the average distance from each candidate to the nearest design point (this is the U criterion)
- the average harmonic mean distance from each candidate to the design

For S-optimality, the following alternative measures of spread are listed:

- the harmonic mean distance from each design point to the nearest other design point (this is the S criterion)
- the average distance from each design point to the nearest other design point

In addition, the OPTEX procedure can create an output data set, as described in “OUTPUT Statement” on page 742 and in “Output Data Sets” on page 769.

ODS Tables

The following table summarizes the ODS tables that you can request with the PROC OPTTEX statement.

Table 24.5. ODS Tables Produced in PROC OPTEX

ODS Table Name	**Description**	**Statement**	**Option**
ClassLevels	Classification variable levels	CLASS	default
FactorRanges	Continuous variable ranges	default	default
BlockDesignEfficiencies	Block design efficiency criteria	BLOCK	default
Efficiencies	Efficiency criteria for all designs	GENERATE	default
Criteria	Efficiency criteria for a single design	EXAMINE	default
Points	Design points	EXAMINE	POINTS
Information	Information matrix XPX	EXAMINE	INFORMATION
Variance	Inverse information matrix inv(XPX)	EXAMINE	VARIANCE
Status	Optimization status	PROC	STATUS
Distances	Distance criteria for all designs	GENERATE	CRITERION=U or S

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/QC® User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1994 pp.

SAS/QC® User's Guide, Version 8

ISBN 1–58025–493–4

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999