SAS/QC® 14.2 User's Guide
The OPTEX Procedure

Chapter 15
The OPTEX Procedure

Contents

Overview: OPTEX Procedure

The OPTEX procedure searches for optimal experimental designs. You specify a set of candidate design points and a linear model, and the procedure chooses points so that the terms in the model can be estimated as efficiently as possible.

Most experimental situations call for standard designs, such as fractional factorials, orthogonal arrays, central composite designs, or Box-Behnken designs. Standard designs have assured degrees of precision and orthogonality that are important for the exploratory nature of experimentation. However, standard designs are not available in some situations, such as the following:

- Not all combinations of the factor levels are feasible.
- The region of experimentation is irregularly shaped.
- Resource limitations restrict the number of experiments that can be performed.
- There is a nonstandard linear or a nonlinear model.

The OPTEX procedure can generate an efficient experimental design for any of these situations.

NOTE: Instead of using PROC OPTEX directly, a more appropriate tool for you might be the ADX Interface. The ADX Interface is designed primarily for engineers and researchers who require a point-and-click solution for the entire experimental process, from building the designs through determining significant effects to optimization and reporting. In addition to offering the standard designs, ADX makes it easy to use PROC OPTEX to find optimal designs for nonstandard factorial, response surface, and mixture experiments, with and without blocking. For more information about the ADX Interface, see *Getting Started with the SAS ADX Interface for Design of Experiments*.

Features

This section summarizes key features of the OPTEX procedure.

The OPTEX procedure offers various criteria for searching a design; these criteria are summarized in Table 15.1 and Table 15.2. In the formulas for these criteria, $\mathbf{X}$ denotes the design matrix, $\mathcal{C}$ the set of candidate points, and $\mathcal{D}$ the set of design points. The default criterion is D-optimality. You can also use the OPTEX procedure to generate G- and I-efficient designs.

The OPTEX procedure also offers a variety of search algorithms, ranging from a simple sequential search (Dykstra 1971) to the computer-intensive Fedorov algorithm (Fedorov 1972; Cook and Nachtsheim 1980). You can customize many aspects of the search, such as the initialization method and the number of iterations.

You can use the full general linear modeling facilities of the GLM procedure to specify a model for your design, allowing for general polynomial effects in addition to classification or ANOVA effects. Optionally, you can specify the following:

- design points to be optimally augmented
- fixed covariates (for example, blocks) for the design
- prior precisions for Bayesian optimal design

The OPTEX procedure is an interactive procedure. After specifying an initial design, you can submit additional statements without reinvoking the OPTEX procedure. Once you have found a design, you can do the following:

- examine the design
- output the design to a data set
- change the model and find another design
- change the characteristics of the search and find another design

Table 15.1 Information-Based Optimality Criteria

Criterion	**Goal**	**Formula**
D-optimality	Maximize determinant of the information matrix	$\max\|\mathbf{X}'\mathbf{X}\|$
A-optimality	Minimize sum of the variances of estimated coefficients	$\min\ \mathrm{trace}(\mathbf{X}'\mathbf{X})^{-1}$

Table 15.2 Distance-Based Optimality Criteria

Criterion	**Goal**	**Formula**
U-optimality	Minimize distance from design to candidates	$\min\sum_{\mathbf{x}\in\mathcal{C}} d(\mathbf{x},\mathcal{D})$
S-optimality	Maximize distance between design points	$\min\sum_{\mathbf{y}\in\mathcal{D}} d(\mathbf{y},\mathcal{D}-\mathbf{y})$

Learning about the OPTEX Procedure

To learn the basic syntax of the OPTEX procedure, read the introductory example in the next section, which covers a typical application of optimal designs. Other applications are illustrated in the section “Optimal Design Scenarios” on page 986. The summary tables in the section “Summary of Functions” on page 990 provides an overview of the syntax. The section “Examples: OPTEX Procedure” on page 1022 illustrates construction of complex designs.

Getting Started: OPTEX Procedure

The examples in this section illustrate basic features of the OPTEX procedure. In addition, the examples show how a variety of SAS software tools can be used to construct candidate sets. If you are working through these examples on your own computer, note that the randomness in the OPTEX procedure's search algorithm will cause your results to be slightly different from those shown.

For illustrations of complex features, see the section "Examples: OPTEX Procedure" on page 1022.

Constructing a Nonstandard Design

NOTE: See *Constructing a Nonstandard Design* in the SAS/QC Sample Library.

This example shows how you can use the OPTEX procedure to construct a design for a complicated experiment for which no standard design is available.

A chemical company is designing a new reaction process. The engineers have isolated the following five factors that might affect the total yield:

Variable	Description	Range
RTemp	Temperature of the reaction chamber	150–350 degrees
Press	Pressure of the reaction chamber	10–30 psi
Time	Amount of time for the reaction	3–5 minutes
Solvent	Amount of solvent used	20–25%
Source	Source of raw materials	1, 2, 3, 4, 5

Although there are only two solvent levels of interest, the reaction control factors (RTemp, Press, and Time) might be curvilinearly related to the total yield, and thus require three levels in the experiment. The Source factor is categorical with five levels. In addition, some combinations of the factors are known to be problematic; simultaneously setting all three reaction control factors to their lowest feasible levels will result in worthless sludge, whereas setting them all to their highest levels can damage the reactor. Standard experimental designs do not apply to this situation.

Creating the Candidate Set

You can use the OPTEX procedure to generate a design for this experiment. The first step in generating an optimal design is to prepare a data set that contains the candidate runs (that is, the feasible factor level combinations). In many cases, this step involves the most work. You can use a variety of SAS data manipulation tools to set up the candidate data set. In this example, the candidate runs are all possible combinations of the factor levels except those in which all three control factors are at their low levels and those in which all three are at their high levels. The PLAN procedure (see *SAS/STAT User's Guide*) provides an easy way to create a full factorial data set, which can then be subsetted by using the DATA step, as shown in the following statements:

```
proc plan ordered;
   factors RTemp=3 Press=3 Time=3 Solvent=2 Source=5 / noprint;
   output out=Candidate
      RTemp   nvals=(150 to 350 by 100)
      Press   nvals=( 10 to  30 by  10)
      Time    nvals=(  3 to   5       )
      Solvent nvals=( 20 to  25 by   5)
      Source  nvals=(  1 to   5       );
data Candidate; set Candidate;
   if (^((RTemp = 150) & (Press = 10) & (Time = 3)));
   if (^((RTemp = 350) & (Press = 30) & (Time = 5)));
run;
proc print data=Candidate(obs=10);
run;
```

A partial listing of the candidate data set Candidate is shown in Figure 15.1.

Figure 15.1 Candidate Set of Runs for Chemical Reaction Design

Obs	RTemp	Press	Time	Solvent	Source
1	150	10	4	20	1
2	150	10	4	20	2
3	150	10	4	20	3
4	150	10	4	20	4
5	150	10	4	20	5
6	150	10	4	25	1
7	150	10	4	25	2
8	150	10	4	25	3
9	150	10	4	25	4
10	150	10	4	25	5

Generating the Design

The next step is to invoke the OPTEX procedure, specifying the candidate data set as the input data set. You must also provide a model for the experiment by using the MODEL statement, which uses the linear modeling syntax of the GLM procedure (see *SAS/STAT User's Guide*). Because Source is a classification (qualitative) factor, you need to specify it in a CLASS statement. To detect possible crossproduct effects in the other factors, in addition to the quadratic effects of the three reaction control factors, you can use a modified response surface model, as shown in the following statements:

```
proc optex data=Candidate seed=12345;
   class Source;
   model Source Solvent|RTemp|Press|Time@2
         RTemp*RTemp Press*Press Time*Time;
run;
```

Note that the MODEL statement does not involve a response variable (unlike the MODEL statement in the GLM procedure). The default number of runs for a design is assumed by the OPTEX procedure to be 10 plus the number of parameters (a total of $10 + 18 = 28$ in this case). Thus, the procedure searches for 28 runs among the candidates in Candidate that enable D-optimal estimation of the effects in the model. (For a precise definition of D-optimality, see the section "Optimality Criteria" on page 1014.) Randomness is built into the search algorithm to overcome the problem of local optima. By default, the OPTEX procedure takes 10 random "tries" to find the best design. The output, shown in Figure 15.2, lists efficiency factors for the 10 designs found. These designs are all very close in terms of their D-efficiency.

Figure 15.2 Efficiencies for Chemical Reaction Design

The OPTEX Procedure

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	57.0082	32.8139	78.3162	0.8319
2	56.7660	27.3874	75.8168	0.8563
3	56.2145	28.7217	74.9937	0.8594
4	55.8960	28.7509	74.4196	0.8559
5	55.7341	29.9372	74.4554	0.8544
6	55.6224	31.4902	73.6200	0.8626
7	55.5762	28.3016	75.8959	0.8652
8	55.5080	30.3889	78.4385	0.8552
9	55.3366	28.5103	74.7014	0.8614
10	55.2176	26.8133	76.2307	0.8660

The final step is to save the best design in a data set. You can do this interactively by submitting the OUTPUT statement immediately after the preceding statements. Then use the PRINT procedure to list the design. The design is listed in Figure 15.3.

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

Figure 15.3 Optimal Design for Chemical Reaction Process Experiment

Obs	Solvent	RTemp	Press	Time	Source
1	20	150	20	4	5
2	20	250	10	5	5
3	20	350	30	3	5
4	25	150	30	5	5
5	25	250	10	3	5
6	25	350	20	5	5
7	20	150	10	5	4
8	20	150	30	3	4
9	20	350	10	3	4
10	20	350	20	5	4
11	25	250	30	4	4
12	20	250	10	3	3
13	20	350	30	4	3
14	25	150	30	3	3
15	25	350	10	5	3
16	25	350	20	3	3
17	20	150	30	5	2
18	20	250	30	3	2
19	20	350	10	5	2
20	25	150	10	4	2
21	25	250	20	5	2
22	25	350	30	4	2
23	20	150	20	3	1
24	20	250	20	4	1
25	20	250	30	5	1
26	25	150	10	5	1
27	25	350	10	4	1
28	25	350	30	3	1

Customizing the Number of Runs

The OPTEX procedure provides options that enable you to customize many aspects of the design optimization process. Suppose the budget for this experiment can accommodate only 25 runs. You can use the N= option in the GENERATE statement to request a design with this number of runs.

```
proc optex data=Candidate seed=12345;
   class source;
   model source Solvent|RTemp|Press|Time@2
         RTemp*RTemp Press*Press Time*Time;
   generate n=25;
run;
```

Including Specific Runs

If there are factor combinations that you want to include in the final design, you can use the OPTEX procedure to *augment* those combinations optimally. For example, suppose you want to force four specific factor combinations to be in the design. If these combinations are saved in a data set, you can force them into the design by specifying the data set with the AUGMENT= option in the GENERATE statement. This technique is demonstrated in the following statements:

```
data Preset;
   input Solvent RTemp Press Time Source;
   datalines;
20 350 10 5 4
20 150 10 4 3
25 150 30 3 3
25 250 10 5 3
;
proc optex data=Candidate seed=12345;
   class Source;
   model Source Solvent|RTemp|Press|Time@2
         RTemp*RTemp Press*Press Time*Time;
   generate n=25 augment=preset;
   output out=Reactor2;
run;
```

The final design is listed in Figure 15.4.

```
proc print data=Reactor2;
run;
```

Figure 15.4 Augmented Design for Chemical Reaction Process Experiment

Obs	Solvent	RTemp	Press	Time	Source
1	20	150	30	3	5
2	20	350	20	5	5
3	25	150	10	4	5
4	25	250	30	4	5
5	20	350	10	5	4
6	20	350	30	3	4
7	25	150	30	5	4
8	25	250	10	3	4
9	25	350	20	5	4
10	20	150	10	4	3
11	20	150	30	5	3
12	20	350	20	3	3
13	25	150	30	3	3
14	25	250	10	5	3
15	20	150	10	5	2
16	20	250	30	5	2
17	20	350	10	4	2
18	25	150	20	3	2
19	25	350	10	5	2
20	20	250	10	3	1
21	20	250	20	4	1
22	20	350	30	4	1
23	25	150	10	5	1
24	25	350	10	3	1
25	25	350	30	3	1

Note that the points in the AUGMENT= data set appear as observations 7, 11, 15, and 16.

Using an Alternative Search Technique

You can also specify a variety of optimization methods by using the GENERATE statement. The default method is relatively fast; although other methods might find better designs, they take longer to run and the improvement is usually only marginal. The method that generally finds the best designs is the Fedorov procedure (Fedorov 1972). The following statements show how to request this method:

```
proc optex data=Candidate seed=12345;
   class Source;
   model Source Solvent|RTemp|Press|Time@2
         RTemp*RTemp Press*Press Time*Time;
   generate n=25 method=fedorov;
   output out=Reactor2;
run;
```

The efficiencies for the resulting designs are shown in Figure 15.5.

Figure 15.5 Efficiency Factors for the Fedorov Search

The OPTEX Procedure

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	56.9072	27.6680	75.2161	0.9023
2	56.8715	27.4939	72.8202	0.9058
3	56.6148	27.7799	75.1840	0.9031
4	56.3021	31.4247	76.0654	0.9044
5	56.0569	25.4498	70.2491	0.9290
6	55.9501	26.8714	75.6991	0.9144
7	55.8461	29.0473	74.1291	0.9138
8	55.8355	26.9242	76.8595	0.9062
9	55.7253	27.4625	74.3391	0.9189
10	55.6071	26.3825	74.1827	0.9107

In this case, the Fedorov procedure takes several times longer than the default method, and D-efficiency shows no improvement. On the other hand, the longer search method often does improve the design and might take only a few seconds on a reasonably fast computer.

Optimal Design Scenarios

The following examples briefly describe some additional common situations that call for optimal designs. These examples show how you can use a variety of SAS software tools to generate an appropriate set of candidate runs and use the OPTEX procedure to search the candidate set for an optimal design.

The emphasis here is on the programming techniques; output is omitted.

Constructing a Saturated Second-Order Design

Suppose you want a design for seven two-level factors that is as small as possible but still permits estimation of all main effects and two-factor interactions—that is, a *saturated* design. Among standard orthogonal arrays, the smallest appropriate 2^k design has 64 runs, far more than the 29 parameters you want to estimate. To generate a D-efficient nonorthogonal design, first use the FACTEX procedure to create the full set of $2^7 = 128$ candidate runs, and then invoke the OPTEX procedure with a full second-order model, asking for a saturated design, as follows:

```
proc factex;
   factors x1-x7;
   output out=Candidate1;
run;
proc optex data=Candidate1 seed=12345;
   model x1|x2|x3|x4|x5|x6|x7@2;
   generate n=saturated;
   output out=Design1a;
run;
```

The default search procedure quickly finds a design with a D-efficiency of 82.3%. If search time is not an issue, you can try a more powerful search technique. For example, you can specify 500 tries with the Fedorov method:

```
proc optex data=Candidate1 seed=12345;
   model x1|x2|x3|x4|x5|x6|x7@2;
   generate n=saturated
            method=fedorov
            iter=500;
   output out=Design1b;
run;
```

This takes much longer to run, and the resulting design is only slightly more D-efficient.

Augmenting a Resolution 4 Design

In a situation similar to the previous example, suppose you have performed an experiment for seven two-level factors with a 16-run, fractional factorial design of resolution 4. You can estimate all main effects with this design, but some two-factor interactions will be confounded with each other. You now want to add enough runs to estimate all two-factor interactions as well. You can use the FACTEX procedure to create the original design in addition to the candidate set.

```
proc factex;
   factors x1-x7;
   output out=Candidate2;
run;
   model resolution=4;
   size design=min;
   output out=Augment2;
run;
```

Now specify Augment2 (the data set that contains the design to be augmented) with the AUGMENT= option in the GENERATE statement:

```
proc optex data=Candidate2 seed=12345;
   model x1|x2|x3|x4|x5|x6|x7@2;
   generate n=30 augment=Augment2;
   output out=Design2;
run;
```

Handling Many Variables

When you have many factors, the set of all possible factor level combinations might be too large to work with as a candidate set. Suppose you want a main-effects design for 15 three-level factors. The complete set of $3^{15} = 14,348,907$ candidates is too large to use with the OPTEX procedure; in fact, it might be too large to store in your computer. One solution is to find a subset of the full factorial set to use as candidates. For example, an alternative candidate set is the 81-run orthogonal design of resolution 3, which can easily be constructed by the FACTEX procedure:

```
proc factex;
   factors x1-x15 / nlev=3;
   model resolution=3;
   size design=81;
   output out=Candidate3;
run;
proc optex data=can3 seed=12345;
   class x1-x15;
   model x1-x15;
   generate n=saturated;
   output out=Design3;
run;
```

Constructing an Incomplete Block Design

An incomplete block design is a design for v (qualitative) treatments in b blocks of size k, where $k < v$ so that not all treatments can occur in each block. To construct an incomplete block design with the OPTEX procedure, simply create a candidate data set that contains a treatment variable with t values and then use the BLOCKS statement. For example, the following statements construct a design for seven treatments in seven blocks of size three:

```
data Candidate4;
   do Treatment = 1 to 7;
      output;
   end;
proc optex data=Candidate4 seed=12345;
   class Treatment;
   model Treatment;
   blocks structure=(7)3;
run;
```

The resulting design is *equireplicated* in the sense that each treatment occurs the same number of times and *balanced* in the sense that each pair of treatments occurs together in the same number of blocks. Balanced designs, when they exist, are known to be optimal, and the OPTEX procedure usually succeeds at finding them for small to moderately sized problems.

Constructing a Mixture-Process Design

Suppose you want to design an experiment with three *mixture factors* X1, X2, and X3 (continuous factors that represent proportions of the components of a mixture) and one *process factor* A (a classification factor with five levels). Furthermore, suppose that X1 can account for no more than 50% of the mixture. The following statements create a data set containing the vertices and generalized edge centroids of the region that is defined by the mixture factor constraints and then use the FACTEX procedure (see the section "Overview: FACTEX Procedure" on page 618) to create a candidate set that includes the process factor:

```
data XVert;
   input x1 x2 x3 @@;
datalines;
0.50 0.000 0.500
0.50 0.500 0.000
0.00 1.000 0.000
0.00 0.000 1.000
0.00 0.500 0.500
0.50 0.250 0.250
0.25 0.000 0.750
0.25 0.750 0.000
0.25 0.375 0.375
;
proc factex;
   factors a / nlev=5;
   output out=Candidate5 pointrep=XVert;
run;
```

Analyzing mixture designs with linear models can be problematic because of the constraint that the mixture factors sum to one; however, to generate an optimal design, you can simply drop one of the mixture factors. The following statements use the preceding candidate set to find an optimal design for fitting the main effect of A and a second-order model in the mixture factors:

```
proc optex data=Candidate5 seed=12345;
   class a;
   model a x1|x2 x1*x1 x2*x2;
run;
```

See Example 15.10 for a more detailed example of a mixture experiment.

Syntax: OPTEX Procedure

The following statements are available in 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 must use the PROC OPTEX and MODEL statements. You can use the other statements as needed. The OPTEX procedure is interactive, so you can 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 immediately 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 as follows:

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

In addition, a CLASS statement that names classification variables must precede the MODEL statement that uses those variables.

Summary of Functions

Table 15.3, Table 15.4, and Table 15.5 classify the OPTEX statements and options by function.

Table 15.3 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 $(\mathbf{X}'\mathbf{X})^{-1}$	GENERATE	CRITERION=A
Maximize $\|\mathbf{X}'\mathbf{X}\|$	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*
Specify CLASS variable parameterization	CLASS	/ PARAM=*method*
Display CLASS variable parameterization	PROC OPTEX	CLASSPARAM
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
Specify initial design	GENERATE	INITDESIGN=*SAS-data-set*

Table 15.4 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 15.5 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 $\mathbf{X}'\mathbf{X}$	EXAMINE	INFORMATION
Specific optimal design	EXAMINE	NUMBER=*design-number*
Variance matrix $(\mathbf{X}'\mathbf{X})^{-1}$	EXAMINE	VARIANCE
Suppress all output	PROC OPTEX	NOPRINT

PROC OPTEX Statement

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

The PROC OPTEX statement invokes the procedure. You can specify the following *options*:

CLASSPARAM
displays a table that summarizes the parameterization of classification variables in the model for the design.

CODING=NONE | STATIC | ORTH | 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. You can specify the following values:

NONE	suppresses coding of effects. This option is equivalent to the NOCODE option.
ORTH	specifies orthogonal coding with respect to the points in the candidate data set and in the AUGMENT= and INITDESIGN= data sets.
ORTHCAN	specifies orthogonal coding with respect to the points in the candidate data set only.
STATIC	requests that the values of all effects be coded to have maximum and minimum values of +1 and –1, respectively.

By default, CODING=STATIC. For more information about coding, see the section "Design Coding" on page 1012. Although CODING=STATIC is the default, CODING=ORTH usually produces give more meaningful 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 more information, see the section "DATA= Data Set" on page 1007.

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, $\epsilon = 0.00001$.

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. By default, NAMELEN=20.

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

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

SEED=*s*
specifies an integer used to start the pseudorandom number generator for initialization (see the section "Search Methods" on page 1017). If you do not specify a seed, or if you specify a value less than or equal to zero, the seed is generated by default from reading the time of day from the computer's clock.

STATUS=*status-level*

requests that the status of the search be checked at the specified *status-level*, which must be 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 is specified in 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 need to specify STATUS=3 or STATUS=4 only very rarely, because evaluating and displaying the status at either of these levels usually makes the search 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 the section "Optimal Blocking" on page 1019.

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 that corresponds 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.

You can specify any of the following three mutually exclusive *block-specifications*:

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

specifies a data set to use in providing a general covariance matrix for the runs, where *variables* names the variables in this data set that contain the columns of the covariance matrix for the runs. For an example, see Example 15.9.

DESIGN=*SAS-data-set*

specifies a data set to use in providing a general covariate model. In addition to this data set, you must use the CLASS and MODEL statements to specify a covariate model. Covariate models are specified in the same way as the treatment model; CLASS and MODEL statements that come after a BLOCKS statement that involves the DESIGN= specification are interpreted as applying to the covariate model. For an example, see Example 15.8.

STRUCTURE=(*b***)** *k*

specifies a block design that has *b* blocks of size *k*. For an example, see Example 15.7.

You can also specify the following *options*:

INIT=RANDOM | CHAIN

specifies the initialization method for constructing the starting design. You can specify the following values:

CHAIN	selects candidate points in the order in which they occur in the original data set.
RANDOM	constructs the starting design by selecting candidates at random without replacement.

By default, INIT=RANDOM.

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. Specify both INIT=CHAIN and ITER=0 to evaluate the initial design itself.

KEEP=*m*

retains only the best *m* designs. 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 *variable* < **(***v-options***)** > < *variable* < **(***v-options***)** > *. . .* > < */ v-options* > > **;**

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

You can specify various *v-options* for each *variable* by enclosing them in parentheses after the variable name. You can also specify global *v-options* for the CLASS statement by placing them after a slash (/). Global *v-options* are applied to all the variables specified in the CLASS statement. However, individual CLASS variable *v-options* override the global *v-options*.

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

NOTE: If you use the DESIGN= option in the BLOCKS statement to specify a data set that contains fixed covariate effects, 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.

DESCENDING

DESC

reverses the sorting order of the classification variable.

ORDER=DATA | FORMATTED | FREQ | INTERNAL

specifies the sorting order for the levels of classification variables. This ordering determines which parameters in the model correspond to each level in the data, so the ORDER= option can be useful when you use the CONTRAST statement. When ORDER=FORMATTED (the default) for numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC OPTEX run or in the DATA step that created the data set), the levels are ordered by their internal (numeric) value. This represents a change from how class levels were ordered before SAS 8, when numeric class levels with no explicit format were ordered by their BEST12. formatted values. In order to revert to the previous ordering, you can specify this format explicitly for the affected classification variables. The change was implemented because the former default behavior for ORDER=FORMATTED often resulted in levels not being ordered numerically. The following table shows how PROC OPTEX interprets values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value (the sort order is machine-dependent)
FREQ	Descending frequency count; levels with the most observations come first in the order
INTERNAL	Unformatted value (the sort order is machine-dependent)

By default, ORDER=FORMATTED.

For more information about sorting order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

PARAM=*method*

specifies the parameterization method for the classification variables. Design matrix columns are created from CLASS variables according to the specified coding scheme.

By default, PARAM=ORTHEFFECT. This represents a change from how classification variables were parameterized before SAS 9, when the default was PARAM=EFFECT. In order to revert to the previous parameterization, you can specify PARAM=EFFECT explicitly for the affected classification variables. The change was implemented because an orthogonal parameterization leads to D- and A-efficiency values that more realistically reflect the true efficiency of the design.

You can specify the following parameterization *methods*, all of which are full rank. The orthogonal versions perform a scaled, intercept-augmented Gram-Schmidt orthogonalization on the columns of the corresponding nonorthogonal parameterizations. Each description shows how a model that has one CLASS variable A with four levels (1, 2, 5, and 7) is coded.

EFFECT specifies effect coding. Three columns are created to indicate group membership of the nonreference levels. The REF= option in the CLASS statement determines the reference level. For the reference level, all three dummy variables have a value of –1. For example, if the reference level is 7 (REF=7), the design matrix columns for A are as follows.

Effect Coding			
A	**Design Matrix**		
1	1	0	0
2	0	1	0
5	0	0	1
7	–1	–1	–1

Parameter estimates of CLASS main effects that use the effect coding scheme estimate the difference in the effect of each nonreference level compared to the average effect over all four levels.

POLYNOMIAL | POLY specifies polynomial coding. Three columns are created. The first represents the linear term (x), the second represents the quadratic term (x^2), and the third represents the cubic term (x^3), where x is the level value. If the CLASS levels are numeric, then the ORDER= option in the CLASS statement is ignored and the internal, unformatted values are used. If the CLASS levels are not numeric, they are translated into 1, 2, 3, . . . according to their sorting order. The design matrix columns for A are as follows.

Polynomial Coding			
A	**Design Matrix**		
1	1	1	1
2	2	4	8
5	5	25	125
7	7	49	343

REFERENCE | REF specifies reference cell coding. Three columns are created to indicate group membership of the nonreference levels. The REF= option in the CLASS statement determines the reference level. For the reference level, all three dummy variables have a value of 0. For example, if the reference level is 7 (REF=7), the design matrix columns for A are as follows.

Reference Coding			
A	**Design Matrix**		
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

Parameter estimates of CLASS main effects that use the reference coding scheme estimate the difference in the effect of each nonreference level compared to the effect of the reference level.

ORDINAL | ORD specifies ordinal (“thermometer”) coding. Three columns are created to indicate group membership in successive collections of levels after the first. For example, the design matrix columns for A are as follows.

Ordinal Coding			
A	**Design Matrix**		
1	0	0	0
2	1	0	0
5	1	1	0
7	1	1	1

Parameter estimates of CLASS main effects that use the ordinal coding scheme estimate the difference in the average effect of each successive collection of levels compared to the effect of the first level.

ORTHEFFECT The columns are obtained by applying the Gram-Schmidt orthogonalization to the mean-centered columns for PARAM=EFFECT and then scaling so that the sum of squares for each column equals the number of levels. The design matrix columns for A are as follows.

Orthogonal Effects Coding			
A	**Design Matrix**		
1	1.414	–0.816	–0.577
2	0	1.633	–0.577
5	0	0	1.732
7	–1.414	–0.816	–0.577

ORTHPOLY specifies orthogonal polynomial coding. The columns are obtained by applying the Gram-Schmidt orthogonalization to the mean-centered columns for PARAM=POLY and then scaling so that the sum of squares for each column equals the number of levels. The design matrix columns for A are as follows.

Orthogonal Polynomial Coding			
A	**Design Matrix**		
1	–1.153	0.907	–0.921
2	–0.734	–0.540	1.473
5	0.524	–1.370	–0.921
7	1.363	1.004	0.368

If the CLASS levels are numeric, then the ORDER= option in the CLASS statement is ignored and the internal, unformatted values are used.

ORTHREF specifies orthogonal reference cell coding. The columns are obtained by applying the Gram-Schmidt orthogonalization to the mean-centered columns for PARAM=REFERENCE and then scaling so that the sum of squares for each column equals the number of levels. The design matrix columns for A are as follows.

Orthogonal Reference Coding			
A	**Design Matrix**		
1	1.732	0	0
2	–0.577	1.633	0
5	–0.577	–0.816	1.414
7	–0.577	–0.816	–1.414

ORTHORDINAL The columns are obtained by applying the Gram-Schmidt orthogonalization to the mean-centered columns for PARAM=ORDINAL, and then scaling so that the sum of squares for each column equals the number of levels. The design matrix columns for A are as follows.

Orthogonal Ordinal Coding			
A	**Design Matrix**		
1	–1.732	0	0
2	0.577	–1.633	0
5	0.577	0.816	–1.414
7	0.577	0.816	1.414

REF=*'level'* **| FIRST | LAST**

specifies the reference level for PARAM=EFFECT or PARAM=REFERENCE. You can specify the following values:

'*level*' specifies the *level* of the variable to use as the reference level. You can specify this value only for an individual *v-option*. You cannot specify this value for a global *v-option*.

FIRST designates the first ordered level as reference.

LAST designates the last ordered level as reference.

By default, REF=LAST.

TRUNCATE

determines class levels by using only up to the first 16 characters of the formatted values of CLASS variables. When formatted values are longer than 16 characters, you can use this option in order to revert to the levels as determined in releases previous to SAS 9.

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 section "Output" on page 1021.) You can specify the following *options* to modify the output:

DESIGN

lists the actual points in the selected design. Designs are ordered by the value of the efficiency criterion that is being optimized. Thus, a *design-number* of 1 (specified in the NUMBER= option) corresponds to the best design found, a *design-number* of 2 corresponds to the second best design, and so on. By default, the first design (NUMBER=1) is examined. You can select a different design to be examined by using the NUMBER= option.

INFORMATION

INFO

I

lists the information matrix $\mathbf{X}'\mathbf{X}$ for the selected design.

NUMBER=*design-number*

selects a design to examine by specifying its *design-number*.

VARIANCE

VAR

V

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

For more information about design efficiencies, see the section "Design Efficiency Measures" on page 1011.

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 by doing the following:

- 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)

You can specify the following *options* 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 generated design. When creating designs, the OPTEX procedure adds points from the DATA= data set (or the last data set created, if the DATA= option 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 in the N= option. For more information, see the section "AUGMENT= Data Set" on page 1007.

CRITERION=D | A | U | S

specifies the optimality criterion used in the search. You can specify any one of the following values:

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 $(\mathbf{X}'\mathbf{X})^{-1}$.

D specifies D-optimality; the optimal design maximizes the determinant $|\mathbf{X}'\mathbf{X}|$ of the information matrix for the design.

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

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

where $\mathcal{D}$ is the set of design points and $N_{\mathcal{D}}$ 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*.

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*.

By default, CRITERION=D. For more information about the different criteria, see the section "Optimality Criteria" on page 1014.

INITDESIGN=SEQUENTIAL | RANDOM | PARTIAL <*m***> |** *SAS-data-set*

specifies a method of obtaining an initial design for the search procedure. You can specify the following values:

SEQUENTIAL specifies an initial design chosen by a sequential search. The design that is produced by this option is the same as the design that is produced by METHOD=SEQUENTIAL. You can use this option with other values of the

METHOD= option to specify a sequential design as the initial design for various search methods. For more information, see the section “Search Methods” on page 1017.

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

PARTIAL<(*m*)>
specifies an initial design by 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 the section “Search Methods” on page 1017.)

You can specify the optional integer *m* to modify the selection of n_r. By default, or if $m = 0$, n_r is randomly chosen between 0 and one less than half the number of parameters in the linear model. 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. For notes on choosing n_r, see Galil and Kiefer (1980).

SAS-data-set
specifies a data set that holds the initial design. Use this option when you have a specific design that you want to improve or when you want to evaluate an existing design. For more information, see the section “INITDESIGN= Data Set” on page 1007.

The default initialization method depends on the search procedure as shown in Table 15.6.

Table 15.6 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 by using a random or partially random method of initialization (see the preceding INITDESIGN= option).

The *n* designs that are found are sorted by their respective efficiencies according to the current optimality criterion (see the CRITERION= option on page 1001). 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 specify the *design-number* in the NUMBER= option in the EXAMINE and OUTPUT statements to display the characteristics of a design or to save a design in a data set.

By default, ITER=10.

KEEP=*m*

retains only the best *m* designs. 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*) > | EXCHANGE < (*k*) > | FEDOROV | M_FEDOROV | SEQUENTIAL

specifies the procedure used to search for the optimal design. You can specify the following values:

DETMAX<(*level*)> uses the DETMAX algorithm of Mitchell (1974a). This algorithm is the best-known and most widely used optimal design search algorithm. The optional *level* specifies the maximum excursion level for the search, where *level* is an integer greater than or equal to 1. The default value for *level* is 4. In general, larger values of *level* result in longer search times.

EXCHANGE<(*k*)> uses the simple exchange method of Mitchell and Miller (1970). 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).

FEDOROV uses the Fedorov algorithm (Fedorov 1972), which seeks the pair $(\mathbf{x}, \mathbf{y})$ of one candidate point and one design point that maximizes $\Delta(\mathbf{x}, \mathbf{y})$ and then switches $\mathbf{x}$ for $\mathbf{y}$ in the design.

M_FEDOROV uses the modified Fedorov algorithm of Cook and Nachtsheim (1980), which 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 $\Delta(\mathbf{x},\mathbf{y})$. 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.

SEQUENTIAL uses the sequential search of Dykstra (1971), which starts with an empty design and adds successive candidate points so that the chosen criterion is optimized at each step. The is the simplest and fastest algorithm.

By default, METHOD=EXCHANGE. From fastest to slowest, the methods are as follows:

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

In general, slower methods result in more efficient designs. Although the default method (METHOD=EXCHANGE) always works relatively quickly, you might want to specify a more reliable method, such as METHOD=M_FEDOROV, when you have a fast computer or a small to moderately sized problem.

For more information about the algorithms, see the section "Search Methods" on page 1017.

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 that has *n* points. Specify N=SATURATED to search for a design whose number of points is equal to the number of 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.

The *variables* must be contained in the DATA= data set, which is specified in the PROC OPTEX statement. They can also be contained in other input data sets. If a *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 *variables* for that observation are transferred to the OUT= data set. For more information, see the section "Input Data Sets" on page 1006.

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 any of the following:

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

The variables that are used to form *effects* in the MODEL statement must be present in all input data sets. For more information about input data sets, see the section "Input Data Sets" on page 1006. For more information about the specification of different types of effects and about how the design matrix is defined with respect to the effects, see the section "Specifying Effects in MODEL Statements" on page 1009.

If you use the DESIGN= option in the BLOCKS statement to specify a data set that contains fixed covariate effects, 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.

You can specify the following *options*:

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 that correspond to groups of effects in the model. Groups of effects in the MODEL statement that have 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 15.6 for an example.

When you specify prior precision values, the information matrix for estimating the linear parameters is $\mathbf{X}'\mathbf{X} + \mathbf{P}$, where $\mathbf{X}$ is the design matrix and $\mathbf{P}$ is a diagonal matrix whose diagonal contains the prior precision values that you specify. Thus, in terms of a prior distribution, the inverses of the prior precision values can be interpreted as prior variances for the linear parameters that correspond to each effect. As an alternative interpretation, note that with orthogonal coding the value of the prior for an effect says approximately how many prior "observations' worth" of information you have for that effect. For more information about orthogonal coding, see the section "Design Coding" on page 1012.

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.

You can specify the following *options*:

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 15.7 for an example.

NUMBER=*design-number* **| DBEST | ABEST | GBEST | VBEST**

specifies how to select the design to output. You can specify the following values:

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. To modify the number of designs created, see the ITER= option.
DBEST	selects the design that has the highest D-efficiency value.
ABEST	selects the design that has the highest A-efficiency value.
GBEST	selects the design that has the highest G-efficiency value.
VBEST	selects the design that has the minimum average standard error for prediction.

By default, NUMBER=1.

The DBEST, ABEST, GBEST, and VBEST 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; for more information, see the section "G- and I-Optimality" on page 1015.

Details: OPTEX Procedure

Input Data Sets

This section discusses the five input data sets for the OPTEX procedure. Three of the data sets provide points to be used to generate the design according to the effects you specify in the MODEL statement. Two other data sets provide points to be 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 is optional and contains points that are guaranteed to 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 15.6 is a schematic diagram of the roles of the DATA=, AUGMENT=, and INITDESIGN= data sets in constructing the design. Figure 15.7 presents the role of the DESIGN= data set for block designs.

Figure 15.6 Choosing from DATA= Points

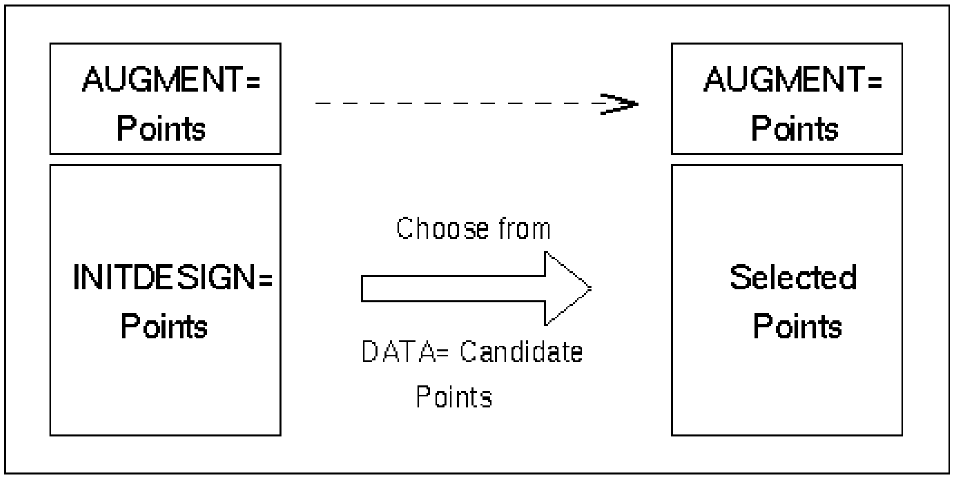
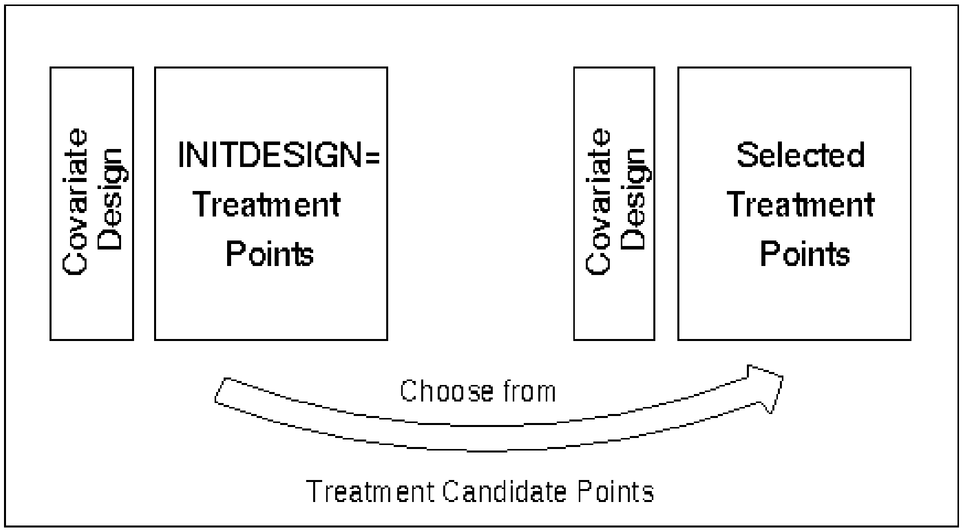


Figure 15.7 Choosing Treatment Candidates


Covariate Design
INITDESIGN= Treatment Points
Covariate Design
Selected Treatment Points
Choose from
Treatment Candidate Points

DATA= Data Set

The DATA= data set provides a set of candidate points to be 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 to be 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 that involve 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 by 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). For an example that uses an AUGMENT= data set, see the section "Including Specific Runs" on page 984.

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 by 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 15.3 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 (you do not have to specify ITER=0 in this case). 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 15.8 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). Because 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 15.9 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 in addition to 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 set. 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.

Because 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.

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

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 *overparameterized* technique for generating the design matrix (see the section "Static Coding" on page 1012.)

Each term in a model, called an *effect*, is a variable or combination of variables. To specify effects, you use a special notation that involves 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 that are 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 list of effect types, assume that A, B, and C are classification variables and X1, X2, and X3 are continuous variables:

- Regressor effects are specified by writing continuous variables by themselves, as follows:

  ```
  X1   X2   X3
  ```

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

- Polynomial effects are specified by joining two or more continuous variables with asterisks, as follows:

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

  Polynomial effects are also referred to as interactions or crossproducts 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 crossproducts.

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

- Main effects are specified by writing classification variables by themselves. as follows:

  ```
  A   B   C
  ```

  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.

Most designs involve main effects because they correspond to the factors in your experiment. For example, in a factorial experiment for a chemical process, the main effects can be metal type, temperature, pressure, and the level of a catalyst.

For information about how the OPTEX procedure generates the $k - 1$ columns in the design matrix that correspond to the main effects of a classification variable, see the section "Design Coding" on page 1012.

- Crossed effects (interactions) are specified by joining class variables with asterisks, as follows:

  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 that correspond to a crossed effect are formed by the horizontal direct products of the constituent main effects.

- Continuous-by-class effects are specified by joining continuous variables and classification variables with asterisks, as follows:

  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 classification variable.

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 by 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 by using rules given by Searle (1971). For example, **A|B|C** is evaluated as follows:

```
A | B | C  →  { A | B } | C
           →  { A  B  A*B } | C
           →  A   B   A*B   C   A*C   B*C   A*B*C
```

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 that it finds in its search for an optimum design:

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

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 percentages) that are required by a hypothetical orthogonal design to achieve the same $|\mathbf{X}'\mathbf{X}|$ and $\text{trace}(\mathbf{X}'\mathbf{X})^{-1}$, respectively (Mitchell 1974b).

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

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

where $\lambda_1, \ldots, \lambda_p$ are the p largest eigenvalues of $\mathbf{A}^{-1}$. If you use the STRUCTURE= block model specification and the treatment model has only one classification variable, 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 might 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. For a definition of the design measures tabulated for these criteria, see the section "Output" on page 1021.

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 that correspond to each model term, turning particular values of the factors into a row vector $\mathbf{x}$. The OPTEX procedure requires a *nonsingular* coding for the design matrix. Because of this, any two coding schemes are related by a nonsingular 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 that correspond to the main effect of a classification variable A are computed as follows: For a design point with A at its ith level, for $1 \le i \le k-1$, the columns of the design matrix associated with A are all 0 except for the ith column, which is 1. When A is at its kth level, all k – 1 columns associated with A are –1. Thus, if α_i denotes the expected response at the ith 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 that correspond to the constituent effects.

This coding corresponds to modeling without *overparameterization*, by using the same method as the CATMOD procedure in SAS/STAT software uses. This is different from the method used by the GLM procedure, which uses an overparameterized 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 $\mathbf{X}_C'\mathbf{X}_C = N_C \cdot \mathbf{I}$, where $\mathbf{X}_C$ is the design matrix for the candidate points, N_C is the number of candidates, and $\mathbf{I}$ is the identity matrix. This is required in order for the D- and A-efficiency measures to make sense. For the CODING=ORTHCAN option, this recoding is accomplished by computing a square matrix $\mathbf{R}$ such that $\mathbf{X}_C'\mathbf{X}_C = \mathbf{R}'\mathbf{R}$ and then transforming each row vector $\mathbf{x}$ as

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

If you specify CODING=ORTH, the recoding is done in a similar fashion, except that the matrix $\mathbf{R}$ is computed according to $\mathbf{X}_C'\mathbf{X}_C + \mathbf{X}_A'\mathbf{X}_A + \mathbf{X}_I'\mathbf{X}_I = \mathbf{R}'\mathbf{R}$, where $\mathbf{X}_A$ and $\mathbf{X}_I$ are the design matrices for the AUGMENT= and INITDESIGN= data sets, respectively (coded as described in the previous section.) Thus, these two orthogonal coding options differ only when there is an AUGMENT= or an INITDESIGN= data set; the CODING=ORTH option includes points from these data sets in computing the orthogonal coding, whereas the CODING=ORTHCAN option uses only the candidates themselves.

Example of Coding

For example, consider a main effect model that has one continuous variable X and one three-level classification variable A. The results of the various coding options are shown in Table 15.7.

Table 15.7 Different Types of Design Coding

Original Data		Design Matrix with CODING=NONE				Design Matrix with CODING=STATIC				Design Matrix with CODING=ORTH			
X	A		X	A1	A2		X	A1	A2		X	A1	A2
1	1	1	1	1	0	1	–1	1	0	1	–1.464	0.598	–0.707
2	2	1	2	0	1	1	–0.6	0	1	1	–0.878	–0.478	1.414
3	3	1	3	–1	–1	1	–0.2	–1	–1	1	–0.293	–1.554	–0.707
4	1	1	4	1	0	1	0.2	1	0	1	0.293	1.554	–0.707
5	2	1	5	0	1	1	0.6	0	1	1	0.878	0.478	1.414
6	3	1	6	–1	–1	1	1	–1	–1	1	1.464	–0.598	–0.707

The first column in each design matrix is an all-ones vector that corresponds 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, because 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 nonsingular transformation A, which changes the determinant of the information matrix as follows:

$$|\mathbf{X}'\mathbf{X}| \quad \rightarrow \quad |\mathbf{A}'\mathbf{X}'\mathbf{X}\mathbf{A}| \;=\; |\mathbf{A}'\mathbf{A}||\mathbf{X}'\mathbf{X}| \;=\; |\mathbf{A}|^2|\mathbf{X}'\mathbf{X}|$$

Thus, recoding simply multiplies the D-criterion by a constant that is the same for all designs. However, 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. However, 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 can also be useful 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 $\mathbf{X}'\mathbf{X}$ 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 $(\mathbf{X}'\mathbf{X})^{-1}$, which is the same as "maximizing" the information $\mathbf{X}'\mathbf{X}$. D- and A-efficiency are different ways of saying how large $(\mathbf{X}'\mathbf{X})$ or $(\mathbf{X}'\mathbf{X})^{-1}$ 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 in the section "Distance-Based Criteria" on page 1016. 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 criteria 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.

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

The determinant is thus a general measure of the size of $(\mathbf{X}'\mathbf{X})^{-1}$. D-optimality is the most commonly used criterion for generating optimal designs and is therefore 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, because each update depends only on the variance of prediction for the current design; see the section "Useful Matrix Formulas" on page 1017.

- $|\mathbf{X}'\mathbf{X}|$ 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.
- $|\mathbf{X}'\mathbf{X}|^{1/p}$ is equal to the geometric mean of the eigenvalues of $\mathbf{X}'\mathbf{X}$.
- The D-optimal design is invariant to nonsingular recoding of the design matrix.

$$|\mathbf{X}'\mathbf{X}| \quad \rightarrow \quad |\mathbf{A}'\mathbf{X}'\mathbf{X}\mathbf{A}| \;=\; |\mathbf{A}'\mathbf{A}||\mathbf{X}'\mathbf{X}| \;=\; |\mathbf{A}|^2|\mathbf{X}'\mathbf{X}|$$

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 $(\mathbf{X}'\mathbf{X})^{-1}$. Like the determinant, the A-optimality criterion is a general measure of the size of $(\mathbf{X}'\mathbf{X})^{-1}$. A-optimality is less commonly used than D-optimality as a criterion for computer optimal design, partly because it is more computationally difficult to update (see the section "Useful Matrix Formulas" on page 1017). Also, A-optimality is *not* invariant to nonsingular 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 $\mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}$. As this formula shows, these two criteria are also related to the information matrix $\mathbf{X}'\mathbf{X}$. 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 the section "Useful Matrix Formulas" on page 1017), 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. Because 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 Nguyen and Piepel (2005)). For more information about specifying the number of tries, see the ITER= option.

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

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

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

Note that both G- and I-optimality are invariant to nonsingular recoding of the design matrix, because 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:

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

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

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

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. Because 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 because all 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 in the sections "Handling Many Variables" on page 988 and "Constructing a Mixture-Process Design" on page 989.

The distance-based criteria (CRITERION=U and CRITERION=S) also require repeated access to the distance between candidate points. PROC OPTEX tries to fit the matrix of these distances in memory; if it cannot, it recomputes them as needed, but this causes 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 particular 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 approximately 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 the section "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 that a faster algorithm finds will be within 1–2% 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 a design and deleting them from this design. If $\mathbf{V} = (\mathbf{X}'\mathbf{X})^{-1}$ is the inverse of the information matrix for the design at any stage, then the change in $\mathbf{V}$ that results from adding a point $\mathbf{x}$ to a design (which adds a new row $\mathbf{x}$ to the design matrix) is

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

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

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

It follows that adding $\mathbf{x}$ multiplies the determinant of the information matrix by $1 + \mathbf{x}'\mathbf{V}\mathbf{x}$. Likewise, deleting $\mathbf{y}$ multiplies the determinant by $1 - \mathbf{y}'\mathbf{V}\mathbf{y}$. For any point $\mathbf{z}$, the quantity $\mathbf{z}'\mathbf{V}\mathbf{z}$ 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. Here, the trace that results from adding a point $\mathbf{x}$ to a design depends on

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

and the trace that results from deleting a point $\mathbf{y}$ to this design 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 and 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 1001). If you specify a data set for an initial design for this search procedure, no search will be made; in this way, you can use the OPTEX procedure to evaluate an existing design.

Because 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.

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 might vary between $N_D + k$ and $N_D - k$, where N_D 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 1003). For the precise stopping rules for each excursion and for the entire search, see Mitchell (1974a). Due to the mentioned excursions, the DETMAX algorithm might not be a good choice when the design you want to construct is saturated or near-saturated.

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—that is, 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.

From the equations in the section "Useful Matrix Formulas" on page 1017 (see also Nguyen and Piepel (2005, sec. 4)), it follows that simultaneously adding a point **x** and deleting a point **y** multiplies the determinant of the information matrix by $1 + \Delta(x, y)$, where:

$$\Delta(\mathbf{x}, \mathbf{y}) = \mathbf{x}'\mathbf{V}\mathbf{x} - \mathbf{y}'\mathbf{V}\mathbf{y} + (\mathbf{x}'\mathbf{V}\mathbf{y})^2 - (\mathbf{x}'\mathbf{V}\mathbf{x})(\mathbf{y}'V\mathbf{y})$$

The quantity $\Delta(\mathbf{x}, \mathbf{y})$ is often referred to as Fedorov's delta function.

At each step, the Fedorov algorithm (Fedorov 1972) seeks the pair $(\mathbf{x}, \mathbf{y})$ of one candidate point and one design point that maximizes $\Delta(\mathbf{x}, \mathbf{y})$ and then switches **x** for **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 **y** in the design with the candidate point **x** that maximizes $\Delta(\mathbf{x},\mathbf{y})$. 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.

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 that have smallest variance in the current design are considered for exchange. You can specify k-exchange as the search procedure for OPTEX by specifying 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.

For a detailed review of the preceding search methods, see Nguyen and Miller (1992).

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 $\mathbf{X}'\mathbf{A}\mathbf{X}$ for a certain symmetric, nonnegative-definite matrix $\mathbf{A}$ that depends on the blocks. The algorithm is based on two kinds of low-rank changes to the treatment design matrix $\mathbf{X}$: *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 $\mathbf{X}'\mathbf{A}\mathbf{X}$ and $|\mathbf{X}'\mathbf{A}\mathbf{X}|$. These update formulas can be generalized to apply whenever the information matrix for the treatment design has the form $\mathbf{X}'\mathbf{A}\mathbf{X}$, not just when $\mathbf{A}$ 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 the GENERATE statement to make the data set Edesign the treatment design:

```
generate initdesign=Edesign method=sequential;
```

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 by using knowledge of the specific design problem. For example, if the default algorithm (METHOD=EXCHANGE) runs quickly but does not clearly indicate 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; see Snee (1985), and see the introductory examples in the sections "Handling Many Variables" on page 988 and "Constructing a Mixture-Process Design" on page 989.

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 1001). 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 is probably best to start each try with a completely random design (INITDESIGN=RANDOM). On the other hand, prior knowledge might 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.

For more information about the efficiencies, see the section "Design Efficiency Measures" on page 1011. 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 the following:

- $\log |\mathbf{X}'\mathbf{X}|$
- $\mathrm{trace}(\mathbf{X}'\mathbf{X})^{-1}$
- 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 PROC OPTEX lists alternative measures of coverage and spread instead of the preceding efficiencies. For U-optimality the following measures are listed:

- 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 the sections "OUTPUT Statement" on page 1005 and "Output Data Sets" on page 1008.

ODS Tables

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

Table 15.8 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

Examples: OPTEX Procedure

Example 15.1: Nonstandard Linear Model

NOTE: See *A Nonstandard Linear Model* in the SAS/QC Sample Library.

This 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 because 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 four cosine variables (c1, c2, c3, and c4) and three sine variables (s1, s2, and s3). The following statements produce Output 15.1.1:

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

Output 15.1.1 Sampling Wildlife Habitats over Time

The OPTEX Procedure

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	19.7379	57.7350	1.3229
4	31.6103	19.3793	57.7350	1.3229
5	31.6103	19.2916	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 15.1.1. 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. Because the OPTEX procedure is interactive, you need to submit only the following statements (immediately after the preceding statements) to produce Output 15.1.2 and Output 15.1.3:

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

proc sort data=d1;
   by  Month Habitat;
run;
proc print data=d1;
   var Month Habitat;
run;

proc sort data=d6;
   by  Month Habitat;
run;
proc print data=d6;
   var Month Habitat;
run;
```

Output 15.1.2 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 15.1.3 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 15.1.2. 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 15.1.3 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 15.1.4:

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

Output 15.1.4 Searching Directly for an A-Efficient Design

The OPTEX Procedure

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard 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 15.2: Comparing the Fedorov Algorithm to the Sequential Algorithm

Note: See *Engine Mapping Problem* in the SAS/QC Sample Library.

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 (Vance 1986). The three control variables AFR (air fuel ratio), EGR (exhaust gas recirculation), and SA (spark advance) and their possible settings are shown in the following table:

Variable	Values						
AFR	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.

Because the factors have different numbers of levels, you can use the PLAN procedure (see *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 AFR=4 ordered EGR=6 ordered SA=8 ordered
            / noprint;
         output out=a
            AFR nvals=(15, 16, 17, 18)
            EGR nvals=(0.020, 0.177, 0.377, 0.566, 0.921, 1.117)
            SA  nvals=(10, 16, 22, 28, 34, 40, 46, 52);
      run;
```

The Fedorov algorithm (Fedorov 1972) is generally the most successful optimal design search algorithm, although it also typically can take relatively much longer to run than other algorithms. This algorithm is not the default search method for the OPTEX procedure. However, you can request that it be used by specifying the METHOD=FEDOROV option in the GENERATE statement. For example, the following statements produce Output 15.2.1:

```
   proc optex data=a seed=61552;
      model AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
      generate n=50 method=fedorov iter=100 keep=10;
   run;
```

Output 15.2.1 Efficiencies with the Fedorov Algorithm

The OPTEX Procedure

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.5246	24.5897	96.3915	0.4231
2	46.5241	24.5901	96.3926	0.4233
3	46.5238	24.5844	96.2306	0.4231
4	46.5237	24.5855	96.2318	0.4233
5	46.5219	24.5866	96.4790	0.4233
6	46.5192	24.5832	96.3070	0.4231
7	46.5192	24.5832	96.3070	0.4231
8	46.5190	24.5741	96.1695	0.4232
9	46.5189	24.5841	96.3062	0.4233
10	46.5188	24.5755	96.3020	0.4234

The Fedorov search method for the preceding problem requires a few seconds for 100 tries on a 2.8GHz desktop PC.

For comparison, you can use the METHOD=SEQUENTIAL option in the GENERATE statement, as shown in the following statements, which produce Output 15.2.2:

```
proc optex data=a seed=33805;
   model AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=sequential iter=100 keep=10;
run;
```

Output 15.2.2 Efficiencies with Sequential Algorithm

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	46.5246	24.5897	96.3915	0.4231
2	46.5241	24.5901	96.3926	0.4233
3	46.5238	24.5844	96.2306	0.4231
4	46.5237	24.5855	96.2318	0.4233
5	46.5219	24.5866	96.4790	0.4233
6	46.5192	24.5832	96.3070	0.4231
7	46.5192	24.5832	96.3070	0.4231
8	46.5190	24.5741	96.1695	0.4232
9	46.5189	24.5841	96.3062	0.4233
10	46.5188	24.5755	96.3020	0.4234

In a fraction of the run time required by the Fedorov method, the sequential algorithm finds a design with a relative D-efficiency of $46.4009/46.5246 = 99.73\%$ compared to the best design found by the Fedorov method, and with *better* A-efficiency. As this demonstrates, if absolute D-optimality is not required, a faster, simpler search might be sufficient.

Example 15.3: Using an Initial Design to Search an Optimal Design

NOTE: See *Engine Mapping Problem* in the SAS/QC Sample Library.

This example is a continuation of Example 15.2.

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 because 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 AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=detmax initdesign=sequential;
run;
```

The results, which are displayed in Output 15.3.1, show an improvement over the sequential design itself (Output 15.2.2) but not over the DETMAX algorithm with the default initialization method (Output 15.2.1).

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 15.3.1 Initializing with a Sequential Design

The OPTEX Procedure

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 might 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 AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=saturated method=detmax;
   output out=b;
run;

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

proc optex data=a seed=33805;
   model AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=detmax initdesign=c;
run;
```

The results are displayed in Output 15.3.2 and Output 15.3.3. The resulting design is 99.9% D-efficient and 98.4% A-efficient relative to the best design found by the straightforward approach (Output 15.2.1), and it takes considerably less time to produce.

Output 15.3.2 Efficiencies for the Unreplicated Saturated Design

The OPTEX Procedure

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 15.3.3 Initializing with a Data Set

The OPTEX Procedure

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

Example 15.4: Optimal Design Using an Augmented Best Design

NOTE: See *Engine Mapping Problem* in the SAS/QC Sample Library.

This example is a continuation of Example 15.2.

You can specify a set of points that you want to be included in the final design that the OPTEX procedure finds by 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 AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=25 method=detmax;
   output out=b;
run;

proc optex data=a seed=37034;
   model AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=detmax augment=b;
run;
```

The result (see Output 15.4.1 and Output 15.4.2) 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 because the run time for the DETMAX algorithm is essentially linear in the size of the design (see the section "Memory and Run-Time Considerations" on page 1016).

Output 15.4.1 Efficiencies for the 25-Point Design to Be Augmented

The OPTEX Procedure

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 15.4.2 Efficiencies for the Augmented 50-Point Design

The OPTEX Procedure

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 15.5: Optimal Design Using a Small Candidate Set

NOTE: See *Engine Mapping Problem* in the SAS/QC Sample Library.

This example is a continuation of Example 15.4.

A well-chosen initial design can speed up the search procedure, as illustrated in Example 15.2. 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 AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=sequential;
   output out=b;
run;
proc freq;
   table AFR EGR SA / nocum;
run;
```

Output 15.5.1 Factor-Level Frequencies for Sequential Design

The FREQ Procedure

AFR	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 15.5.1, 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 by using 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 15.5.2):

```
proc plan;
   factors AFR=4 ordered EGR=4 ordered SA=4 ordered
           / noprint;
   output out=a AFR nvals=(15, 16, 17, 18)
                EGR nvals=(0.020, 0.377, 0.566, 1.117)
                SA  nvals=(10, 28, 34, 52);
run;
proc optex seed=61552;
   model AFR|EGR|SA@2 AFR*AFR EGR*EGR SA*SA;
   generate n=50 method=detmax;
run;
```

Output 15.5.2 Optimal Design Using a Smaller Candidate Set

The OPTEX Procedure

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

The resulting design is about as good as the best one obtained from a complete candidate set ($>$ 99.9% relative D-efficiency and marginally higher relative A-efficiency) and takes much less time to find.

For another example of reducing the candidate set for the optimal design search, see the section "Handling Many Variables" on page 988.

Example 15.6: Bayesian Optimal Design

NOTE: See *Bayesian Optimal Design* 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 option 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=Candidates;
run;

proc optex data=Candidates 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=Design;
run;
```

With orthogonal coding, the value of the prior for an effect indicates approximately 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 indicate that there is no prior information for the main effects and 16 runs' worth of information for each two-factor interaction. For more information about orthogonal coding, see the section "Design Coding" on page 1012.

The efficiencies are shown in Output 15.6.1.

Output 15.6.1 Efficiencies for Bayesian Optimal Designs

The OPTEX Procedure

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 3 tries out of 10. It might be a good idea to repeat the search with more tries (see the ITER= option). You can use the ALIASING option of the GLM procedure to list the aliasing structure for the design:

```
data Design; set Design;
   y = ranuni(654231);
proc glm data=Design;
   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 15.6.2. Most of the main effects are indeed unconfounded with two-factor interactions, although many two-factor interactions are confounded with each other.

Output 15.6.2 Aliasing Structure for Bayesian Optimal Design

The GLM 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 15.7: Balanced Incomplete Block Design

NOTE: See *Balanced Incomplete Block Design* in the SAS/QC Sample Library.

This example uses the BLOCKS statement to construct a balanced incomplete block design (BIBD). 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 Candidates;
   do Treatment = 1 to 7;
      do Block = 1 to 7;
         output;
      end;
   end;
run;
```

```
proc optex data=Candidates seed=8327 coding=orth;
   class Treatment Block;
   model Treatment Block;
   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 15.7.1.

Output 15.7.1 Efficiency Factors for $v = b = 7$, $k = 3$ Designs

The OPTEX Procedure

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

Because 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 10 tries. A more serious limitation is that this approach sometimes fails to find a design that has 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 preceding incomplete block design problem. In this case, the candidate set simply consists of the seven treatment levels.

```
data Candidates;
   do Treatment = 1 to 7;
      output;
   end;
run;

proc optex data=Candidates seed=73462 coding=orth;
   class Treatment;
   model Treatment;
   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 STRUCTURE= option in the BLOCKS statement), the efficiency of each design as an incomplete block design is also listed (Output 15.7.2).

Output 15.7.2 Efficiency Factors for $v = b = 7$, k = 3 Optimal Blocking Designs

The OPTEX Procedure

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.

Because 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=Block;
proc print data=BIBD;
run;
```

The final design is shown in Output 15.7.3.

Output 15.7.3 Balanced Incomplete Block Design for $v = b = 7$, $k = 3$

Obs	BLOCK	Treatment
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

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%.

Example 15.8: Optimal Design with Fixed Covariates

NOTE: See *Optimal Design with Fixed Covariates* 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 use the DESIGN= option in the BLOCKS statement to specify the data set that contains the covariates. Covariate models are specified in the same way as the treatment model.

This 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 that is thought to be related to the response has been recorded for each of the experimental units. For example, if the treatments are different types of animal feed, a typical covariate might be the initial weight of each animal. The following statements create the data sets Cov and Treatment, which contain 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;
0.46 0.54 0.58 0.60 0.73 0.77 0.82 0.84 0.89 0.95
;
data Treatment;
   do t = 1 to 5; output; end;
run;
proc optex data=Treatment seed=17364 coding=orthcan;
   class t;
   model t;
   blocks design=Cov;
   model u u*u;
   output out=Design;
run;

proc print data=Design;
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 the following:

- 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, whereas CLASS and MODEL statements that come after a BLOCKS statement that involves the DESIGN= blocks specification are interpreted as applying to the covariate model.

Output 15.8.1 shows the listing of the efficiency values for the 10 designs that are found. Note that the efficiencies are the same for all tries. A listing of the design is shown in Output 15.8.2.

Output 15.8.1 Optimal Treatment Efficiency Factors with a Quadratic Covariate Effect

The OPTEX Procedure

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 15.8.2 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 that ignores the blocks is first generated, and the result is used as the treatment set for optimal blocking. This enables 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 Harville;
   input t @@;
   datalines;
1   2   3   4   5   1   2   3   4   5
;
proc optex data=Treatment coding=orthcan;
   class t;
   model t;
   generate initdesign=Harville 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 15.8.3. They are the same as for the design found by the OPTEX procedure.

Output 15.8.3 Treatment Efficiency Factors for Harville's Design

The OPTEX Procedure

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 relabeling 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.

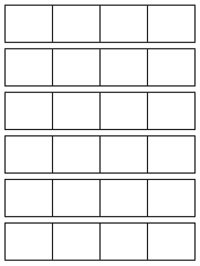
Example 15.9: Optimal Design in the Presence of Covariance

NOTE: See *Optimal Design in Presence of Covariance* in the SAS/QC Sample Library.

The BLOCKS statement finds a design that maximizes the determinant $|\mathbf{X}'\mathbf{A}\mathbf{X}|$ of the treatment information matrix, where $\mathbf{A}$ depends on the block or covariate model. Alternatively, you can directly specify the matrix $\mathbf{A}$ to find the D-optimal design when $\mathbf{A}$ 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 that correspond 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 $\mathbf{A}$.

For example, suppose you want to compare the effects of seven different fertilizers on crop yield, by using seven long, narrow blocks of four plots each, as depicted in Figure 15.8.

Figure 15.8 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

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

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 $\mathbf{X}'\mathbf{A}\mathbf{X}$, where

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

In this formula, $\mathbf{V}$ is the block diagonal matrix of the plot-by-plot correlation structure, with seven copies of R_4 on the diagonal. The matrix $\mathbf{Z}$ is the design matrix that corresponds to the block effect. The optimal design should take into account this neighbor covariance structure in addition to the block structure.

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

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

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 use the COVAR= option in the BLOCKS statement to specify the data set A as the covariance matrix for the runs, as follows:

```
data Fertilizer;
   do f = 1 to 7; output; end;
run;
proc optex data=Fertilizer 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 all var {f};
   NBD = shape(f,7,4);
   print NBD [format=2.];
```

These PROC IML statements read in the selected levels of fertilizer and the reshape them into seven 4-run blocks before printing them. The resulting design is shown in Output 15.9.1. 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 occurS equally often on horizontally adjacent plots.

Output 15.9.1 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 15.10: Adding Space-Filling Points to a Design

NOTE: See *Adding Space-filling Points to a Design* in the SAS/QC Sample Library.

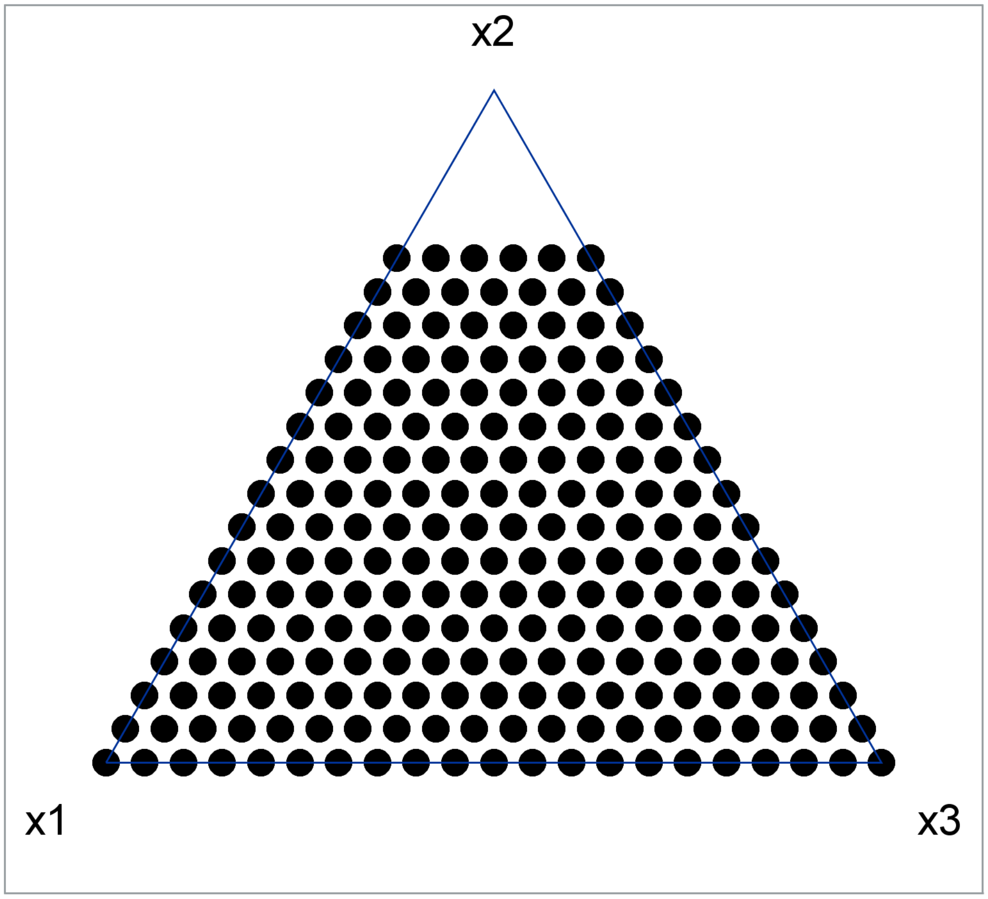
Suppose you want a 15-run experiment for three mixture factors x1, x2, and x3; furthermore, suppose that x3 cannot account for any more than 75% of the mixture. The vertices and generalized edge centroids of the region defined by these constraints make up a good candidate set to use with the OPTEX procedure for finding a D-optimal design for such an experiment. 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 that are chosen according to U-optimality, which seeks to cover the candidate region as well as possible.

The following statements create a candidate data set that contains 216 points in the region that is defined by the constraints $x_1 + x_2 + x_3 = 1$ and $x_3 \leq 0.75$ on the factors:

```
data a;
   do x1 = 0 to 100 by 5;
      do x3 = 0 to 100 by 5;
         x2 = 100 - x1 - x3;
         if (0<= x2 <= 75) then output;
      end;
   end;
run;
data a; set a;
   x1 = x1 / 100;
   x2 = x2 / 100;
   x3 = x3 / 100;
run;
```

The constraint that the factor levels sum to 1 means that the candidate points all lie on a plane. Thus, the values of all three variables can be displayed in a two-dimensional "mixture plot," as shown in Output 15.10.1.

Output 15.10.1 Points in the Feasible Region for Constrained Mixture Design

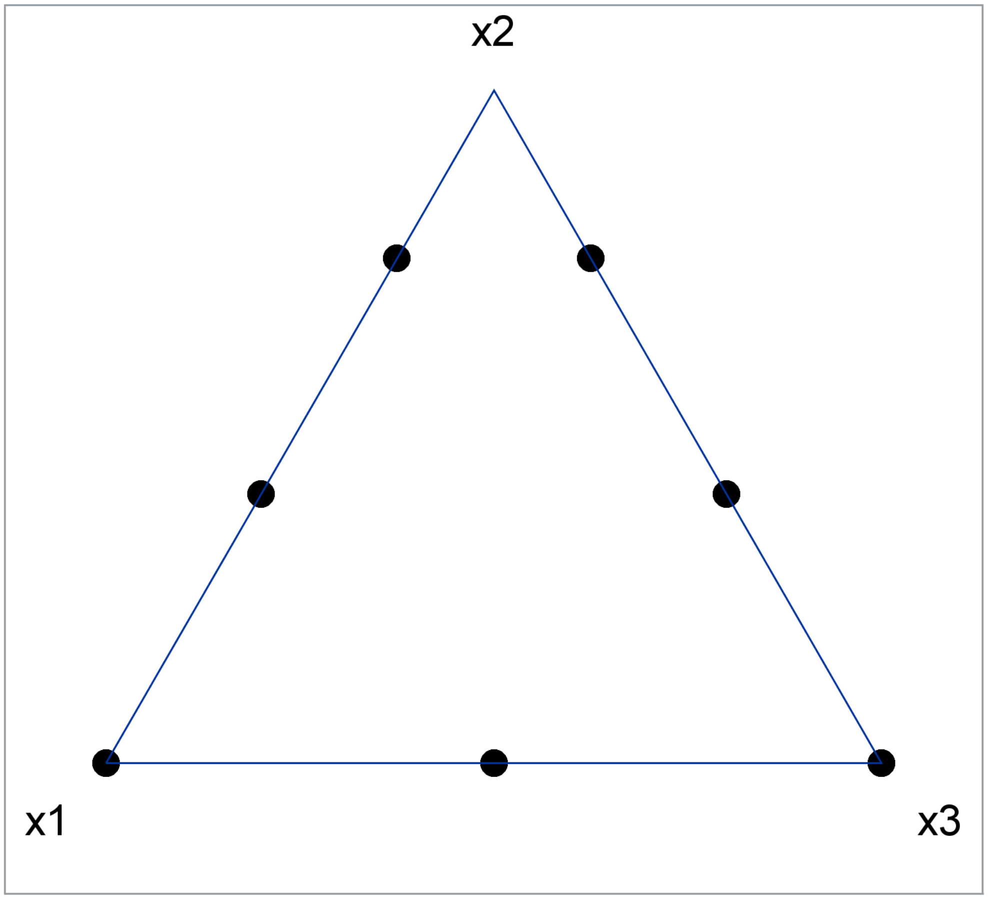


You can use the OPTEX procedure to select 10 points from the mentioned candidate points optimal for estimating a second-order model in the mixture factors:

```
proc optex data=a seed=60868 nocode;
   model x1|x2|x3@2 / noint;
   generate n=10;
   output out=b;
run;
```

The resulting points are plotted in Output 15.10.2. There are only seven unique points, indicating that the D-optimal design replicates some chosen candidate points.

Output 15.10.2 D-Optimal Constrained Mixture Design

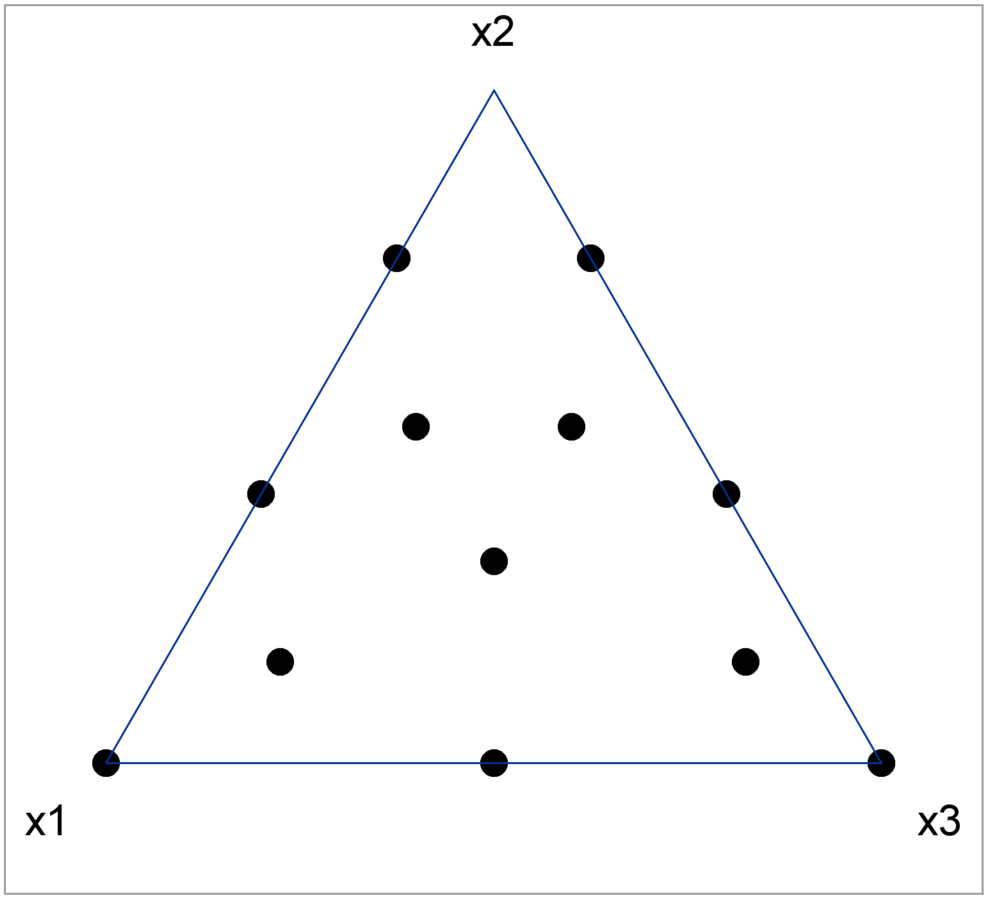


The D-optimal design leaves a large "hole" in the feasible region. The following statements "fill in the hole" in the optimal design that is saved in B by augmenting it with points chosen from the candidate data set a to optimize the U-criterion:

```
proc optex data=a seed=4321 nocode;
   model x1 x2 x3 / noint;
   generate n=15 augment=b criterion=u;
   output out=c;
run;
```

The resulting points are shown in Output 15.10.3. 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 that uses the OPTEX procedure agrees with visual intuition for this small problem. This indicates that the general approach will yield an appropriate design for higher-dimensional problems that cannot be visualized.

Output 15.10.3 D-optimal Constrained Mixture Design Filled in U-optimally



References

Cochran, W. G., and Cox, G. M. (1957). *Experimental Designs*. 2nd ed. New York: John Wiley & Sons.

Cook, R. D., and Nachtsheim, C. J. (1980). "A Comparison of Algorithms for Constructing Exact D-Optimal Designs." *Technometrics* 22:315–324.

Cook, R. D., and Nachtsheim, C. J. (1989). "Computer-Aided Blocking of Factorial and Response-Surface Designs." *Technometrics* 31:339–346.

DuMouchel, W., and Jones, B. (1994). "A Simple Bayesian Modification of D-Optimal Designs to Reduce Dependence on an Assumed Model." *Technometrics* 36:37–47.

Dykstra, O., Jr. (1971). "The Augmentation of Experimental Data to Maximize $|\mathbf{X}'\mathbf{X}|$." *Technometrics* 13:682–688.

Fedorov, V. V. (1972). *Theory of Optimal Experiments.* Translated and edited by W. J. Studden and E. M. Klimko. New York: Academic Press.

Galil, Z., and Kiefer, J. (1980). "Time- and Space-Saving Computer Methods, Related to Mitchell's DETMAX, for Finding D-Optimum Designs." *Technometrics* 22:301–313.

Harville, D. A. (1974). "Nearly Optimal Allocation of Experimental Units Using Observed Covariate Values." *Technometrics* 16:589–599.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). "Minimax and Maximin Distance Designs." *Journal of Statistical Planning and Inference* 26:131–148.

Johnson, M. E., and Nachtsheim, C. J. (1983). "Some Guidelines for Constructing Exact D-Optimal Designs on Convex Design Spaces." *Technometrics* 25:271–277.

Mitchell, T. J. (1974a). "An Algorithm for the Construction of D-Optimal Experimental Designs." *Technometrics* 16:203–210.

Mitchell, T. J. (1974b). "Computer Construction of 'D-Optimal' First-Order Designs." *Technometrics* 20:211–220.

Mitchell, T. J., and Miller, F. L., Jr. (1970). *Use of Design Repair to Construct Designs for Special Linear Models*. Mathematics Division Annual Progress Report ORNL-4661, Oak Ridge National Laboratory.

Nguyen, N.-K., and Miller, A. J. (1992). "A Review of Exchange Algorithms for Constructing Discrete D-Optimal Designs." *Computational Statistics and Data Analysis* 14:489–498.

Nguyen, N.-K., and Piepel, G. F. (2005). "Computer-Generated Experimental Designs for Irregular-Shaped Regions." *Quality Technology and Quantitative Management* 2:147–160.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Snee, R. D. (1985). "Computer-Aided Design of Experiments—Some Practical Experiences." *Journal of Quality Technology* 17:222–236.

Vance, L. C. (1986). *Computer Construction of Experimental Designs*. General Motors Research Report GMR-5411, General Motors Laboratories, Warren, MI.

Subject Index

Syntax Index