Response Surface Methodology, Stat 579 Homework 07 Name: Answer Key
Fall 2014 Prof. Erik B. Erhardt

Part I. (80 points) I recommend reading through all the parts of the HW (with my adjustments) before starting; this may save you some work.
MMA-RSM Chapter 7: 7.1, 7.2, 7.3, 7.5, 7.7, 7.25.

- For 7.7, confirm that the problem statement is correct. If there is an error/are errors, tell what it is/they are and give the correct solution. (There may not be an error.)
- For 7.25, do this for $k = 6(\frac{1}{2}\text{rep})$ instead of $k = 5(\frac{1}{2}\text{rep})$.

General: Try to do all calculations in R. All R code for the assignment should be included with the part of the problem it addresses (for code and output use a fixed-width font, such as Courier). Code is used to calculate result; text is used to report and interpret results – do not report or interpret results in the code.

(15pts) **1. 7.1** Consider the design of an experiment to fit the first-order response model $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \varepsilon$. The design used is a 2^{5-2} fractional factorial with defining relations $I = ABC = x_1x_2x_3$ and $I = CDE = x_3x_4x_5$. The factors are quantitative, and thus we use the notation x_1, x_2, x_3, ..., rather than A, B, C, ...

(a) (5 pts) Construct the matrix $\mathbf{X}$ for the first-order model using the design described above.

Solution: The design matrix $\mathbf{X}$ for the fractional factorial design, 2^{5-2}_{III} using $I = x_1x_2x_3 = x_3x_4x_5 = x_1x_2x_4x_5$, is given below. It was constructed by defining x_1, x_2 and x_4 in Yate's order, defining $x_3 = x_1x_2$ then $x_5 = x_3x_4$, and verifing that $x_5 = x_1x_2x_4$.

```
design.7.1 <- cube( ~ x1 + x2 + x4
                  , generators = c(x3 ~ x1 * x2, x5 ~ x3 * x4)
                  , n0 = 0
                  , randomize = FALSE)
design.7.1
```

```
##   run.order std.order x1.as.is x2.as.is x4.as.is x3.as.is x5.as.is
## 1         1         1       -1       -1       -1        1       -1
## 2         2         2        1       -1       -1       -1        1
## 3         3         3       -1        1       -1       -1        1
## 4         4         4        1        1       -1        1       -1
## 5         5         5       -1       -1        1        1        1
## 6         6         6        1       -1        1       -1       -1
## 7         7         7       -1        1        1       -1       -1
## 8         8         8        1        1        1        1        1
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ x1.as.is
## x2 ~ x2.as.is
## x4 ~ x4.as.is
## x3 ~ x3.as.is
## x5 ~ x5.as.is
```

(b) (5 pts) Is the design orthogonal? Explain why or why not.

Solution: The design is orthogonal because $\mathbf{X}^\top\mathbf{X} = 8\mathbf{I}_6$ is a diagonal matrix.

```
X.7.1 <- as.matrix(design.7.1[,c("x1", "x2", "x3", "x4", "x5")])
X.7.1 <- cbind(1, X.7.1)
t(X.7.1) %*% X.7.1
```

```
##      x1 x2 x3 x4 x5
##    8  0  0  0  0  0
## x1 0  8  0  0  0  0
## x2 0  0  8  0  0  0
## x3 0  0  0  8  0  0
## x4 0  0  0  0  8  0
## x5 0  0  0  0  0  8
```

(c) (5 pts) Is the design a variance-optimal design — that is, one that results in minimum values of $[N\text{Var}(b_i)]/\sigma^2$. Note that the weight N is used in order to take sample size into account.

Solution: Since this is an orthogonal design with points at ± 1 (the extremes), it is variance-optimal.

(25 pts) **2. 7.2** Consider the situation of Exercise 7.1. Suppose we use four center runs to augment the 2^{5-2} design constructed.

(a) (5 pts) What is the advantage of using the center runs?

Solution: Without altering orthogonality, center runs allow for tests of curvature and lack-of-fit. Also, there are now $4 - 1$ df to estimate pure-error, and therefore can run tests for regression coefficients.

(b) (5 pts) Will the resulting 12-run design be orthogonal? Explain.

Solution: Adding center points doesn't alter orthogonality. Vectors of 0 magnitude are by definition orthogonal to all vectors.

(c) (5 pts) Will the resulting design be variance-optimal?

Solution: Since not all points are at the extremes (center points), the design is no longer variance optimal.

(d) (5 pts) If your answer to (c) is no, give a 12-run first-order design with values of $\text{Var}(b_i)/\sigma^2$ smaller than the 12-run design with center runs.

Solution: First, we assess the variance (via the covariance matrix) of the design in 7.2(a).

```
design.7.2a <- cube( ~ x1 + x2 + x4
                   , generators = c(x3 ~ x1 * x2, x5 ~ x3 * x4)
                   , n0 = 4
                   , randomize = FALSE)
design.7.2a
```

```
##    run.order std.order x1.as.is x2.as.is x4.as.is x3.as.is x5.as.is
## 1          1         1       -1       -1       -1        1       -1
## 2          2         2        1       -1       -1       -1        1
## 3          3         3       -1        1       -1       -1        1
## 4          4         4        1        1       -1        1       -1
## 5          5         5       -1       -1        1        1        1
## 6          6         6        1       -1        1       -1       -1
## 7          7         7       -1        1        1       -1       -1
## 8          8         8        1        1        1        1        1
## 9          9         9        0        0        0        0        0
## 10        10        10        0        0        0        0        0
## 11        11        11        0        0        0        0        0
## 12        12        12        0        0        0        0        0
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ x1.as.is
## x2 ~ x2.as.is
## x4 ~ x4.as.is
## x3 ~ x3.as.is
## x5 ~ x5.as.is
```

```
X.7.2a <- as.matrix(design.7.2a[,c("x1", "x2", "x3", "x4", "x5")])
X.7.2a <- cbind(1, X.7.2a)
VarBeta.By.Sigma2.7.2a <- solve(t(X.7.2a) %*% X.7.2a)
```

	V1	x1	x2	x3	x4	x5
	0.083	0.000	0.000	0.000	0.000	0.000
x1	0.000	0.125	0.000	0.000	0.000	0.000
x2	0.000	0.000	0.125	0.000	0.000	0.000
x3	0.000	0.000	0.000	0.125	0.000	0.000
x4	0.000	0.000	0.000	0.000	0.125	0.000
x5	0.000	0.000	0.000	0.000	0.000	0.125

Instead of 4 center runs, I add 4 runs which are the runs 5–8 using $x_3 = -x_1x_2$, which are 4 runs of a fold-over on $x_3 = x_1x_2$. The result is not orthogonal.

```
design.7.2d <- cube( ~ x1 + x2 + x4
                   , generators = c(x3 ~ x1 * x2, x5 ~ x3 * x4)
                   , n0 = 0
                   , randomize = FALSE)
design.7.2d <- as.data.frame(rbind(design.7.2d, design.7.2d[5:8,]))
design.7.2d[9:12, "x3"] <- -design.7.2d[9:12, "x3"]
design.7.2d[9:12, "x5"] <- -design.7.2d[9:12, "x5"]
X.7.2d <- as.matrix(design.7.2d[,c("x1", "x2", "x3", "x4", "x5")])
X.7.2d <- cbind(1, X.7.2d)
VarBeta.By.Sigma2.7.2d <- solve(t(X.7.2d) %*% X.7.2d)
```

	V1	x1	x2	x3	x4	x5
	0.094	0.000	0.000	0.000	−0.031	0.000
x1	0.000	0.083	0.000	0.000	0.000	0.000
x2	0.000	0.000	0.083	0.000	0.000	0.000
x3	0.000	0.000	0.000	0.094	0.000	−0.031
x4	−0.031	0.000	0.000	0.000	0.094	0.000
x5	0.000	0.000	0.000	−0.031	0.000	0.094

(e) (5 pts) Give the variances of regression coefficients for both designs — that is, the 2^{5-2}_{III} with four center runs and your design in (d).

Solution: The variance of the regression coefficients for the 4 center runs are

$$(0.083, 0.125, 0.125, 0.125, 0.125, 0.125).$$

The variance for the 4 runs I added are

$$(0.09375, 0.083, 0.083, 0.09375, 0.09375, 0.09375).$$

While the variance for the intercept is less with the 4 center runs, the variance for the coefficients of effects are all less with my 4 runs.

(10^{pts}) **3. 7.3** Consider again the design constructed in Exercise 7.1. Suppose the fitted model is first-order and the analysis reveals significant lack of fit and a large effect for the interaction $x_1x_4 = x_2x_5$. As a result, a second phase would suggest an augmentation of the design. Suppose the second phase is a fold-over of the design in Exercise 7.1. Is the resulting design orthogonal and thus variance-optimal for the model

$$y_u = \beta_0 + \sum_{i=1}^{5} \beta_i x_{ui} + \sum_{i<j=2}^{5} \beta_{ij} x_{ui} x_{uj} + \varepsilon_u, \quad u = 1, \ldots, 16.$$

Explain why or why not.

Solution:
The original defining relation was $I = x_1x_2x_3$ and $I = x_3x_4x_5$. The fold-over uses $I = -x_1x_2x_3$ and $I = -x_3x_4x_5$. The resulting 2^{5-1}_{IV} has the single generator $I = x_1x_2x_4x_5$, which is not optimal in terms of confounding.
The table below gives the 8 fold-over runs augmenting the design. The $\mathbf{X}$ is defined below that for main effects and crossproduct terms. The $\mathbf{X}^\top\mathbf{X}$ is not diagonal, therefore the design is not orthogonal.

```
design.7.3 <- cube( ~ x1 + x2 + x4
                  , generators = c(x3 ~ x1 * x2, x5 ~ x3 * x4)
                  , n0 = 0
                  , randomize = FALSE)
#design.7.3

# fold-over (reversing sign of x3)
design.7.3.foldover <- foldover(design.7.3, variables = c("x3"), randomize = FALSE)
```

```
#design.7.3.foldover

# join together
design.7.3.all <- djoin(design.7.3, design.7.3.foldover)
design.7.3.all
```

```
##    run.order std.order x1.as.is x2.as.is x4.as.is x3.as.is x5.as.is Block
## 1          1         1       -1       -1       -1        1       -1     1
## 2          2         2        1       -1       -1       -1        1     1
## 3          3         3       -1        1       -1       -1        1     1
## 4          4         4        1        1       -1        1       -1     1
## 5          5         5       -1       -1        1        1        1     1
## 6          6         6        1       -1        1       -1       -1     1
## 7          7         7       -1        1        1       -1       -1     1
## 8          8         8        1        1        1        1        1     1
## 9          1         1       -1       -1       -1       -1       -1     2
## 10         2         2        1       -1       -1        1        1     2
## 11         3         3       -1        1       -1        1        1     2
## 12         4         4        1        1       -1       -1       -1     2
## 13         5         5       -1       -1        1       -1        1     2
## 14         6         6        1       -1        1        1       -1     2
## 15         7         7       -1        1        1        1       -1     2
## 16         8         8        1        1        1       -1        1     2
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ x1.as.is
## x2 ~ x2.as.is
## x4 ~ x4.as.is
## x3 ~ x3.as.is
## x5 ~ x5.as.is
```

```
X.7.3 <- as.matrix(design.7.3.all[,c("x1", "x2", "x3", "x4", "x5")])
X.7.3 <- cbind(1, X.7.3)

# create X with crossproduct terms
for (i in 2:5) {
  for (j in (i+1):6) {
    X.ij <- X.7.3[,i] * X.7.3[,j];
    X.7.3 <- cbind(X.7.3, X.ij);
  }
}

# check orthogonality (yes, if diagonal matrix)
t(X.7.3) %*% X.7.3
```

```
##         x1 x2 x3 x4 x5 X.ij X.ij X.ij X.ij X.ij X.ij X.ij X.ij X.ij X.ij
##      16  0  0  0  0  0    0    0    0    0    0    0    0    0    0    0
## x1    0 16  0  0  0  0    0    0    0    0    0    0    0    0    0    0
## x2    0  0 16  0  0  0    0    0    0    0    0    0    0    0    0    0
## x3    0  0  0 16  0  0    0    0    0    0    0    0    0    0    0    0
## x4    0  0  0  0 16  0    0    0    0    0    0    0    0    0    0    0
## x5    0  0  0  0  0 16    0    0    0    0    0    0    0    0    0    0
## X.ij  0  0  0  0  0  0   16    0    0    0    0    0    0    0    0   16
## X.ij  0  0  0  0  0  0    0   16    0    0    0    0    0    0    0    0
## X.ij  0  0  0  0  0  0    0    0   16    0    0    0   16    0    0    0
## X.ij  0  0  0  0  0  0    0    0    0   16    0   16    0    0    0    0
## X.ij  0  0  0  0  0  0    0    0    0    0   16    0    0    0    0    0
## X.ij  0  0  0  0  0  0    0    0    0   16    0   16    0    0    0    0
## X.ij  0  0  0  0  0  0    0    0   16    0    0    0   16    0    0    0
## X.ij  0  0  0  0  0  0    0    0    0    0    0    0    0   16    0    0
## X.ij  0  0  0  0  0  0    0    0    0    0    0    0    0    0   16    0
## X.ij  0  0  0  0  0  0   16    0    0    0    0    0    0    0    0   16
```

```
# function to determin whether a matrix is orthogonal
f.is.orth <- function(X) {
  return(all(diag(diag(t(X) %*% X)) == (t(X) %*% X)))
}
f.is.orth(X.7.3)
```

```
## [1] FALSE
```

(10pts) **4. 7.5** Consider a situation involving seven design variables when, in fact, a first-order model is required but only eight runs can be used. Design a saturated eight-run design that is variance-optimal.

Solution: A 2^{7-4}_{III} design that is variance-optimal uses generators $I = ABD = ACE = BCF = ABCG$, as given in Table 4.11 on p. 156. Thus, put a, b, and c in Yate's order and use $D = AB$, $E = AC$, $F = BC$, and $G = ABC$, as in the table below. This is orthogonal, with $\mathbf{X}^\top\mathbf{X} = 8\mathbf{I}_8$.

```
design.7.5 <- cube( ~ a + b + c
                  , generators = c(d ~ a * b, e ~ a * c, f ~ b * c, g ~ a * b * c)
                  , n0 = 0
                  , randomize = FALSE)
design.7.5
```

```
##   run.order std.order a.as.is b.as.is c.as.is d.as.is e.as.is f.as.is g.as.is
## 1         1         1      -1      -1      -1       1       1       1      -1
## 2         2         2       1      -1      -1      -1      -1       1       1
## 3         3         3      -1       1      -1      -1       1      -1       1
## 4         4         4       1       1      -1       1      -1      -1      -1
## 5         5         5      -1      -1       1       1      -1      -1       1
## 6         6         6       1      -1       1      -1       1      -1      -1
## 7         7         7      -1       1       1      -1      -1       1      -1
## 8         8         8       1       1       1       1       1       1       1
##
## Data are stored in coded form using these coding formulas ...
## a ~ a.as.is
## b ~ b.as.is
## c ~ c.as.is
## d ~ d.as.is
## e ~ e.as.is
## f ~ f.as.is
## g ~ g.as.is
```

```
# check ortogonality
X.7.5 <- as.matrix(design.7.5[, c("a", "b", "c", "d", "e", "f", "g")])
X.7.5 <- cbind(1, X.7.5)
t(X.7.5) %*% X.7.5
```

```
##     a b c d e f g
##   8 0 0 0 0 0 0 0
## a 0 8 0 0 0 0 0 0
## b 0 0 8 0 0 0 0 0
## c 0 0 0 8 0 0 0 0
## d 0 0 0 0 8 0 0 0
## e 0 0 0 0 0 8 0 0
## f 0 0 0 0 0 0 8 0
## g 0 0 0 0 0 0 0 8
```

(10pts) **5. 7.7** Consider the "model inadequacy" material in Section 7.2. The material can be used to nicely illustrate the aliasing ideas discussed in Chapter 4. Aliasing plays an important role when one deals with fractional factorials for fitting first-order and first-order-plus-interaction response surface models. Suppose one fits a first-order model using a 2^{3-1}_{III} with defining relation $I = x_1x_2x_3$.
However, suppose the true model is

$$\text{E}[y] \quad = \quad \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 + \beta_{12}x_1x_2 + \beta_{13}x_1x_3 + \beta_{23}x_2x_3.$$

Using Equation 7.2, we obtain ...
For 7.7, confirm that the problem statement is correct. If there is an error/are errors, tell what it is/they are and give the correct solution. (There may not be an error.)

Solution: The five errors in the book are

1. the defining relation $I = x_1x_2x_3$ does not give the $\mathbf{X}_1$ matrix defined. Instead, the defining relation used is $I = -x_1x_2x_3$. Therefore,

2. $\mathbf{X}_1$ should have a -1 in the last row for x_3, and
3. $\mathbf{X}_2$ should have two -1s on the last row. The expression for $\mathrm{E}[\hat{\beta}_1]$
4. should have a subscript 1 on the $\mathbf{X}$, and
5. should have $\underset{\sim}{\beta}_2$ in place of $\underset{\sim}{y}$.

Given these corrections, the $\mathbf{A}$ is given in the output below, which gives the proposed expected values for $\underset{\sim}{\beta}_1$.

```
#### 7.7
X1 <- matrix(c( 1, -1,  1,  1,
                1,  1, -1,  1,
                1,  1,  1, -1,
                1, -1, -1, -1
       ), ncol=4, byrow=TRUE);
X2 <- matrix(c(-1, -1,  1,
               -1,  1, -1,
                1, -1, -1,
                1,  1,  1
       ), ncol=3, byrow=TRUE);

# check orthogonality
A <- solve( t(X1) %*% X1 ) %*% t(X1) %*% X2;
A
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0   -1
## [3,]    0   -1    0
## [4,]   -1    0    0
```

(10pts) **6. 7.25** Consider Table 7.5. The designs suggested in the table are those that block orthogonally and are rotatable or near-rotatable. For example, consider the design under the column headed $k = 5(\frac{1}{2}\text{rep})$. Construct the design completely, and verify that it meets both conditions for orthogonal blocking. For 7.25, do this for $k = 6(\frac{1}{2}\text{rep})$ instead of $k = 5(\frac{1}{2}\text{rep})$.

Solution: We use `ccd.pick()` central composite design generator to provide design options for a 2^6 design. We'll select a design corresponding to a 1/2 rep with orthogonal blocking. We then need to verify the two conditions:

1. That each block is orthogonal.
2. That the SS from each block is proportional to the block size.

By inspection, the full design generated based on aliasing blocks as suggested in Table 3.20 p. 120 has a full factorial in (f, e, d, b), with blocks indexed by (a, c). A half fraction can be selected by retaining blocks (1 and 3) or (2 and 4), since those pairs are effectively a foldover on the blocking variables (a, c).

```
library(rsm)
# specifing the conditions for the 6(1/2-rep) in Tab 7.5
ccd.pick(k = 6, n.c = 2^4, n0.c = 4, blks.c = 2, n0.s = 2)
```

```
##   n.c n0.c blks.c n.s n0.s bbr.c wbr.s bbr.s  N alpha.rot alpha.orth
## 1  16    4      2  12    2     1     1     1 54     2.378      2.366
```

```
# full design, remove two blocks for fractional design
design.7.25 <- ccd( y ~ a + b + c + d + e + f
                  #, generators = f ~ a * b * c * d * e
                  , blocks = ~ c(a * b * c * f, c * d * e * f)  # Table 3.20 p.120
                  , n0 = c(4, 2), randomize = FALSE)
design.7.25
```

```
##      run.order std.order a.as.is b.as.is c.as.is d.as.is e.as.is f.as.is  y Block
## 5            1         1  -1.000  -1.000   1.000  -1.000  -1.000  -1.000 NA     1
## 8            2         2   1.000   1.000   1.000  -1.000  -1.000  -1.000 NA     1
## 10           3         3   1.000  -1.000  -1.000   1.000  -1.000  -1.000 NA     1
## 11           4         4  -1.000   1.000  -1.000   1.000  -1.000  -1.000 NA     1
```

```
## 18          5        5   1.000  -1.000  -1.000  -1.000   1.000  -1.000 NA     1
## 19          6        6  -1.000   1.000  -1.000  -1.000   1.000  -1.000 NA     1
## 29          7        7  -1.000  -1.000   1.000   1.000   1.000  -1.000 NA     1
## 32          8        8   1.000   1.000   1.000   1.000   1.000  -1.000 NA     1
## 33          9        9  -1.000  -1.000  -1.000  -1.000  -1.000   1.000 NA     1
## 36         10       10   1.000   1.000  -1.000  -1.000  -1.000   1.000 NA     1
## 46         11       11   1.000  -1.000   1.000   1.000  -1.000   1.000 NA     1
## 47         12       12  -1.000   1.000   1.000   1.000  -1.000   1.000 NA     1
## 54         13       13   1.000  -1.000   1.000  -1.000   1.000   1.000 NA     1
## 55         14       14  -1.000   1.000   1.000  -1.000   1.000   1.000 NA     1
## 57         15       15  -1.000  -1.000  -1.000   1.000   1.000   1.000 NA     1
## 60         16       16   1.000   1.000  -1.000   1.000   1.000   1.000 NA     1
## 17         17       17   0.000   0.000   0.000   0.000   0.000   0.000 NA     1
## 181        18       18   0.000   0.000   0.000   0.000   0.000   0.000 NA     1
## 191        19       19   0.000   0.000   0.000   0.000   0.000   0.000 NA     1
## 20         20       20   0.000   0.000   0.000   0.000   0.000   0.000 NA     1
## 51          1        1   1.000  -1.000   1.000  -1.000  -1.000  -1.000 NA     2
## 81          2        2  -1.000   1.000   1.000  -1.000  -1.000  -1.000 NA     2
## 101         3        3  -1.000  -1.000  -1.000   1.000  -1.000  -1.000 NA     2
## 111         4        4   1.000   1.000  -1.000   1.000  -1.000  -1.000 NA     2
## 182         5        5  -1.000  -1.000  -1.000  -1.000   1.000  -1.000 NA     2
## 192         6        6   1.000   1.000  -1.000  -1.000   1.000  -1.000 NA     2
## 291         7        7   1.000  -1.000   1.000   1.000   1.000  -1.000 NA     2
## 321         8        8  -1.000   1.000   1.000   1.000   1.000  -1.000 NA     2
## 331         9        9   1.000  -1.000  -1.000  -1.000  -1.000   1.000 NA     2
## 361        10       10  -1.000   1.000  -1.000  -1.000  -1.000   1.000 NA     2
## 461        11       11  -1.000  -1.000   1.000   1.000  -1.000   1.000 NA     2
## 471        12       12   1.000   1.000   1.000   1.000  -1.000   1.000 NA     2
## 541        13       13  -1.000  -1.000   1.000  -1.000   1.000   1.000 NA     2
## 551        14       14   1.000   1.000   1.000  -1.000   1.000   1.000 NA     2
## 571        15       15   1.000  -1.000  -1.000   1.000   1.000   1.000 NA     2
## 601        16       16  -1.000   1.000  -1.000   1.000   1.000   1.000 NA     2
## 171        17       17   0.000   0.000   0.000   0.000   0.000   0.000 NA     2
## 1811       18       18   0.000   0.000   0.000   0.000   0.000   0.000 NA     2
## 1911       19       19   0.000   0.000   0.000   0.000   0.000   0.000 NA     2
## 201        20       20   0.000   0.000   0.000   0.000   0.000   0.000 NA     2
## 52          1        1   1.000  -1.000  -1.000  -1.000  -1.000  -1.000 NA     3
## 82          2        2  -1.000   1.000  -1.000  -1.000  -1.000  -1.000 NA     3
## 102         3        3  -1.000  -1.000   1.000   1.000  -1.000  -1.000 NA     3
## 112         4        4   1.000   1.000   1.000   1.000  -1.000  -1.000 NA     3
## 183         5        5  -1.000  -1.000   1.000  -1.000   1.000  -1.000 NA     3
## 193         6        6   1.000   1.000   1.000  -1.000   1.000  -1.000 NA     3
## 292         7        7   1.000  -1.000  -1.000   1.000   1.000  -1.000 NA     3
## 322         8        8  -1.000   1.000  -1.000   1.000   1.000  -1.000 NA     3
## 332         9        9   1.000  -1.000   1.000  -1.000  -1.000   1.000 NA     3
## 362        10       10  -1.000   1.000   1.000  -1.000  -1.000   1.000 NA     3
## 462        11       11  -1.000  -1.000  -1.000   1.000  -1.000   1.000 NA     3
## 472        12       12   1.000   1.000  -1.000   1.000  -1.000   1.000 NA     3
## 542        13       13  -1.000  -1.000  -1.000  -1.000   1.000   1.000 NA     3
## 552        14       14   1.000   1.000  -1.000  -1.000   1.000   1.000 NA     3
## 572        15       15   1.000  -1.000   1.000   1.000   1.000   1.000 NA     3
## 602        16       16  -1.000   1.000   1.000   1.000   1.000   1.000 NA     3
## 172        17       17   0.000   0.000   0.000   0.000   0.000   0.000 NA     3
## 1812       18       18   0.000   0.000   0.000   0.000   0.000   0.000 NA     3
## 1912       19       19   0.000   0.000   0.000   0.000   0.000   0.000 NA     3
## 202        20       20   0.000   0.000   0.000   0.000   0.000   0.000 NA     3
## 53          1        1  -1.000  -1.000  -1.000  -1.000  -1.000  -1.000 NA     4
## 83          2        2   1.000   1.000  -1.000  -1.000  -1.000  -1.000 NA     4
## 103         3        3   1.000  -1.000   1.000   1.000  -1.000  -1.000 NA     4
## 113         4        4  -1.000   1.000   1.000   1.000  -1.000  -1.000 NA     4
## 184         5        5   1.000  -1.000   1.000  -1.000   1.000  -1.000 NA     4
## 194         6        6  -1.000   1.000   1.000  -1.000   1.000  -1.000 NA     4
## 293         7        7  -1.000  -1.000  -1.000   1.000   1.000  -1.000 NA     4
## 323         8        8   1.000   1.000  -1.000   1.000   1.000  -1.000 NA     4
## 333         9        9  -1.000  -1.000   1.000  -1.000  -1.000   1.000 NA     4
## 363        10       10   1.000   1.000   1.000  -1.000  -1.000   1.000 NA     4
## 463        11       11   1.000  -1.000  -1.000   1.000  -1.000   1.000 NA     4
```

```
## 473         12        12  -1.000   1.000  -1.000   1.000  -1.000   1.000 NA     4
## 543         13        13   1.000  -1.000  -1.000  -1.000   1.000   1.000 NA     4
## 553         14        14  -1.000   1.000  -1.000  -1.000   1.000   1.000 NA     4
## 573         15        15  -1.000  -1.000   1.000   1.000   1.000   1.000 NA     4
## 603         16        16   1.000   1.000   1.000   1.000   1.000   1.000 NA     4
## 173         17        17   0.000   0.000   0.000   0.000   0.000   0.000 NA     4
## 1813        18        18   0.000   0.000   0.000   0.000   0.000   0.000 NA     4
## 1913        19        19   0.000   0.000   0.000   0.000   0.000   0.000 NA     4
## 203         20        20   0.000   0.000   0.000   0.000   0.000   0.000 NA     4
## 811          1         1  -2.366   0.000   0.000   0.000   0.000   0.000 NA     5
## 821          2         2   2.366   0.000   0.000   0.000   0.000   0.000 NA     5
## 831          3         3   0.000  -2.366   0.000   0.000   0.000   0.000 NA     5
## 84           4         4   0.000   2.366   0.000   0.000   0.000   0.000 NA     5
## 85           5         5   0.000   0.000  -2.366   0.000   0.000   0.000 NA     5
## 86           6         6   0.000   0.000   2.366   0.000   0.000   0.000 NA     5
## 87           7         7   0.000   0.000   0.000  -2.366   0.000   0.000 NA     5
## 88           8         8   0.000   0.000   0.000   2.366   0.000   0.000 NA     5
## 89           9         9   0.000   0.000   0.000   0.000  -2.366   0.000 NA     5
## 90          10        10   0.000   0.000   0.000   0.000   2.366   0.000 NA     5
## 91          11        11   0.000   0.000   0.000   0.000   0.000  -2.366 NA     5
## 92          12        12   0.000   0.000   0.000   0.000   0.000   2.366 NA     5
## 93          13        13   0.000   0.000   0.000   0.000   0.000   0.000 NA     5
## 94          14        14   0.000   0.000   0.000   0.000   0.000   0.000 NA     5
##
## Data are stored in coded form using these coding formulas ...
## a ~ a.as.is
## b ~ b.as.is
## c ~ c.as.is
## d ~ d.as.is
## e ~ e.as.is
## f ~ f.as.is
```

1. *Show each block is orthogonal.*
For each block, $\mathbf{X}^\top\mathbf{X}$ is diagonal, therefore they are orthogonal.

```
# check orthogonality
X.7.25 <- as.matrix(design.7.25[, c("a", "b", "c", "d", "e", "f")])
X.7.25 <- cbind(1, X.7.25)
# t(X.7.25[(design.7.25£Block == 1),]) %*% X.7.25[(design.7.25£Block == 1),]
# t(X.7.25[(design.7.25£Block == 2),]) %*% X.7.25[(design.7.25£Block == 2),]
# t(X.7.25[(design.7.25£Block == 3),]) %*% X.7.25[(design.7.25£Block == 3),]
# t(X.7.25[(design.7.25£Block == 4),]) %*% X.7.25[(design.7.25£Block == 4),]
# t(X.7.25[(design.7.25£Block == 5),]) %*% X.7.25[(design.7.25£Block == 5),]
f.is.orth(X.7.25[(design.7.25$Block == 1),])
```

```
## [1] TRUE
```

```
f.is.orth(X.7.25[(design.7.25$Block == 2),])
```

```
## [1] TRUE
```

```
f.is.orth(X.7.25[(design.7.25$Block == 3),])
```

```
## [1] TRUE
```

```
f.is.orth(X.7.25[(design.7.25$Block == 4),])
```

```
## [1] TRUE
```

```
f.is.orth(X.7.25[(design.7.25$Block == 5),])
```

```
## [1] TRUE
```

2. *Show that the SS from each block is proportional to the block size.*
We need to verify that our α is what is given on p. 329 in equation (7.33). Let F =number of points in the factorial portion, F_0 =number of added center points in factorial portion, a_0 =number of center points in axial block, and k = the number of factors. Then,

$$\alpha \;=\; \sqrt{\frac{F(2k+a_0)}{2(F+F_0)}} \;=\; \sqrt{\frac{2^{6-1}(2(6)+2)}{2(2^{6-1}+8)}} \;=\; 2\sqrt{7/5} \;=\; 2.366432.$$

This is exactly the value in the axial block (block 3). Therefore we have shown condition 2.
Therefore, this design meets the conditions for orthogonal blocking.
Here's one of the two possible final designs (based on blocks 1 and 3, the other uses blocks 2 and 4):

```
design.7.25.final <- subset(design.7.25, ((Block == 1) | (Block == 3) | (Block == 5)))
rownames(design.7.25.final) <- 1:dim(design.7.25.final)[1]
#design.7.25.final
```

	run.order	std.order	a	b	c	d	e	f	y	Block
1	1	1	−1.000	−1.000	1.000	−1.000	−1.000	−1.000		1
2	2	2	1.000	1.000	1.000	−1.000	−1.000	−1.000		1
3	3	3	1.000	−1.000	−1.000	1.000	−1.000	−1.000		1
4	4	4	−1.000	1.000	−1.000	1.000	−1.000	−1.000		1
5	5	5	1.000	−1.000	−1.000	−1.000	1.000	−1.000		1
6	6	6	−1.000	1.000	−1.000	−1.000	1.000	−1.000		1
7	7	7	−1.000	−1.000	1.000	1.000	1.000	−1.000		1
8	8	8	1.000	1.000	1.000	1.000	1.000	−1.000		1
9	9	9	−1.000	−1.000	−1.000	−1.000	−1.000	1.000		1
10	10	10	1.000	1.000	−1.000	−1.000	−1.000	1.000		1
11	11	11	1.000	−1.000	1.000	1.000	−1.000	1.000		1
12	12	12	−1.000	1.000	1.000	1.000	−1.000	1.000		1
13	13	13	1.000	−1.000	1.000	−1.000	1.000	1.000		1
14	14	14	−1.000	1.000	1.000	−1.000	1.000	1.000		1
15	15	15	−1.000	−1.000	−1.000	1.000	1.000	1.000		1
16	16	16	1.000	1.000	−1.000	1.000	1.000	1.000		1
17	17	17	0.000	0.000	0.000	0.000	0.000	0.000		1
18	18	18	0.000	0.000	0.000	0.000	0.000	0.000		1
19	19	19	0.000	0.000	0.000	0.000	0.000	0.000		1
20	20	20	0.000	0.000	0.000	0.000	0.000	0.000		1
21	1	1	1.000	−1.000	−1.000	−1.000	−1.000	−1.000		3
22	2	2	−1.000	1.000	−1.000	−1.000	−1.000	−1.000		3
23	3	3	−1.000	−1.000	1.000	1.000	−1.000	−1.000		3
24	4	4	1.000	1.000	1.000	1.000	−1.000	−1.000		3
25	5	5	−1.000	−1.000	1.000	−1.000	1.000	−1.000		3
26	6	6	1.000	1.000	1.000	−1.000	1.000	−1.000		3
27	7	7	1.000	−1.000	−1.000	1.000	1.000	−1.000		3
28	8	8	−1.000	1.000	−1.000	1.000	1.000	−1.000		3
29	9	9	1.000	−1.000	1.000	−1.000	−1.000	1.000		3
30	10	10	−1.000	1.000	1.000	−1.000	−1.000	1.000		3
31	11	11	−1.000	−1.000	−1.000	1.000	−1.000	1.000		3
32	12	12	1.000	1.000	−1.000	1.000	−1.000	1.000		3
33	13	13	−1.000	−1.000	−1.000	−1.000	1.000	1.000		3
34	14	14	1.000	1.000	−1.000	−1.000	1.000	1.000		3
35	15	15	1.000	−1.000	1.000	1.000	1.000	1.000		3
36	16	16	−1.000	1.000	1.000	1.000	1.000	1.000		3
37	17	17	−0.000	0.000	−0.000	0.000	0.000	0.000		3
38	18	18	−0.000	0.000	−0.000	0.000	0.000	0.000		3
39	19	19	−0.000	0.000	−0.000	0.000	0.000	0.000		3
40	20	20	−0.000	0.000	−0.000	0.000	0.000	0.000		3
41	1	1	−2.366	0.000	0.000	0.000	0.000	0.000		5
42	2	2	2.366	0.000	0.000	0.000	0.000	0.000		5
43	3	3	0.000	−2.366	0.000	0.000	0.000	0.000		5
44	4	4	0.000	2.366	0.000	0.000	0.000	0.000		5
45	5	5	0.000	0.000	−2.366	0.000	0.000	0.000		5
46	6	6	0.000	0.000	2.366	0.000	0.000	0.000		5
47	7	7	0.000	0.000	0.000	−2.366	0.000	0.000		5
48	8	8	0.000	0.000	0.000	2.366	0.000	0.000		5
49	9	9	0.000	0.000	0.000	0.000	−2.366	0.000		5
50	10	10	0.000	0.000	0.000	0.000	2.366	0.000		5
51	11	11	0.000	0.000	0.000	0.000	0.000	−2.366		5
52	12	12	0.000	0.000	0.000	0.000	0.000	2.366		5
53	13	13	0.000	0.000	0.000	0.000	0.000	0.000		5
54	14	14	0.000	0.000	0.000	0.000	0.000	0.000		5