Robustness of Newly-Proposed Clustered Design

Daniel Burmaster

May 2021

Abstract

In this project, we will propose a new robust design called the "clustered design" consisting of clusters of design points near those of the classical optimal design. We will look at the effectiveness of the clustered design when fitting a simple-linear model, quadratic model, first-order multiple linear regression (MLR), and in some extrapolation scenarios.

For interpolation, we will compare the proposed design with D-optimality, where we take the determinant of the MSE of the ordinary least-squares (OLS) estimate, as our optimality criterion. We compare the clustered design with several other commonly used designs, namely the classical D-optimal design (CDD), Huber's robust design (HRD) (Huber 1975) and the uniform design. We will consider cases where the contamination function is fairly simple (only one higher order term missing in the fitted model) and also when the contamination function is more complex.

Additionally, the clustered design will be tested in some extrapolation scenarios, since robustness is required. This could be important for analyses such as Accelerated Life Testing (ALT) where extrapolation is utilized. Here we will instead adapt Q-optimality for efficiency comparison at a couple select extrapolation points. The clustered design will be compared to a few other designs that are used for extrapolation, such as the Hoel-Levine design (Hoel and Levine 1964), Weins-Xu design (Wiens and Xu 2008), and additionally the uniform design. We will also compare the select designs using a variety of other measurements, such as relative bias, coverage percentage (for empirical and asymptotic confidence intervals and prediction intervals), and simulated standard error.

Acknowledgements

We would like to acknowledge NSERC for funding the support for the research.

Contents

Chapter 1

Introduction

1.1 Problem of Interest

When analyzing data, there are many methods of building models for estimation and prediction. However, no model is ever perfect. Design of Experiments (DoE) is a tool used to improve the efficiency of model-estimation before the experiment has even been run by optimizing the location of the design points within a given design space S.

There are several ways to optimize an estimator $\hat{\boldsymbol{\beta}} = (\beta_1, \beta_2, ..., \beta_p)^T$ during the DoE stage. The classical method of doing this is to find the design such that a scalar function of the covariance of the estimator $Cov(\hat{\boldsymbol{\beta}})$ is minimized. This works excellently if we are very confident in our knowledge of the true model. For example, if we know that the relationship between two factors is simple-linear, then it is well known that the most efficient DoE is to assign half of the sample size to each of the two end points of the design interval. However, what if the initial assumption of the true model wrong? For example, if the true model in reality was a quadratic curve and we fit only a simple-linear model to the data. We may have a low variance in the estimate of our model, but the bias will be significantly high. Therefore, if we are uncertain what the true model is, we will need to take that uncertainty into consideration during the DoE stage. This is why designs are model-specific and require robustness.

Definition: *Robust* means insensitive to violation of assumptions. When constructing a DoE we generally need to assume that the data follows a specific mean response function. A robust design is less affected by getting this assumption wrong. This is a desirable characteristic for a DoE since no model is perfect, therefore we will always assume the model to be something that doesn't completely represent the true model.

A useful value to consider in this scenario is the estimator's mean-squared error which takes into account both the variance and the bias of the estimator:

$MSE(\hat{\boldsymbol{\beta}}) = Cov(\hat{\boldsymbol{\beta}}) + Bias(\hat{\boldsymbol{\beta}})Bias(\hat{\boldsymbol{\beta}})^T$. Through this project $\hat{\boldsymbol{\beta}}$ will be calculated using the OLS method. In the case where we have multiple parameters of estimation, the $MSE(\hat{\boldsymbol{\beta}})$ will be a square matrix, so we can take any proper scalar function of the matrix (for example, the determinant of $MSE(\hat{\boldsymbol{\beta}})$ or the trace) to use it for comparing several designs. Taking into account the variance and bias of the parameter estimates we can analyze the robustness of different methods of constructing the design.

In this paper, the most robust design will be determined by minimizing the determinant of the MSE matrix of $\hat{\boldsymbol{\beta}}$. A design that minimizes determinant is called the D-optimal Robust Design. This paper will compare some popular designs with the newly proposed clustered design using D-optimality.

1.2 Designs of Study and Notations

Notation of Fitted Models Let the true mean response be represented by the function $f(\boldsymbol{x})$ and the fitted one by $f^*(\boldsymbol{x})$. Then the true model is $Y = f(\boldsymbol{x}) + \epsilon$ and the fitted model is $Y^* = f^*(\boldsymbol{x}) + \epsilon$. The notation used to describe the model being fitted will also be defined by the same notation used by Heo (1998, 2001). Therefore, the experimenter can fit the general linear model

$$E(Y|\boldsymbol{x}) = f^*(\boldsymbol{x}) = \boldsymbol{z}^T(\boldsymbol{x})\boldsymbol{\beta}$$

to the data. If the suspected true model of the data is $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, then $\boldsymbol{z}^T(x) = (1, x)^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1)$. We will denote each model by its $\boldsymbol{z}^T(\boldsymbol{x})$ for simplicity. We will number the following models which are to be considered in this simulation:

- Model 1: $\boldsymbol{z}^T(x) = (1, x)^T$
- Model 2: $\boldsymbol{z}^T(x) = (1, x, x^2)^T$
- Model 3: $\boldsymbol{z}^T(\boldsymbol{x}) = (1, x_1, x_2)^T$

Notation of Contamination Space Since the model that we fit to the data is almost never perfect, denote function $g(\boldsymbol{x})$ as the contamination function which is the departure from the assumed model. Therefore, the true mean response function is $f(\boldsymbol{x}) = f^*(\boldsymbol{x}) + g(\boldsymbol{x})$. Huber (1975) puts a limit on the size of the contamination space called the L_2 neighbourhood which is defined as the set of all functions in a class G such that:

$$G = \{g(\boldsymbol{x}) | \int_S g^2(\boldsymbol{x})d\boldsymbol{x} \leq \eta^2, \int_S g(\boldsymbol{x}) * \boldsymbol{z}(\boldsymbol{x})d\boldsymbol{x} = \boldsymbol{0}\}$$

where η^2 is an overall L_2 boundary for the amount of contamination.

Design Notation A general DoE has the form:

$$\xi = \begin{pmatrix} x_1 & x_2 & ... & x_m \\ w_1 & w_2 & ... & w_m \end{pmatrix}$$

where x_i's are the design support points, w_i's are the weights we give to each support point such that $0 < w_i \leq 1$ and $\sum_{i=1}^{m} w_i = 1$ and m is the number of distinct support points. If the total sample size is n, then the number of samples taken at x_i is $n_i = w_i n$.

For simplicity, in this simulation we take design space S to be $[-0.5, 0.5]$.

Classical D-Optimal Design (CDD) The CDD is found such that the determinant of $Cov(\boldsymbol{\beta})$ alone is minimized. For example, when the fitted model is

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$

then the CDD has $m = 3$ support points; and if the design space is $[-0.5, 0.5]$ then a CDD design for this model would be:

$$\xi_{CDD}^{(3)} = \begin{pmatrix} -0.5 & 0 & 0.5 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

Huber's Robust Design (HRD) Huber (1975) proposed a more robust DoE in [5] where the HRD design has a density function $d(x)$ and each design point $x_j, j = 1, ..., m$ is calculated such that (in the case where design space is $[-0.5, 0.5]$):

$$\int_{-0.5}^{x_j} d(x)dx = \frac{j - 0.5}{m} \tag{1.1}$$

Note that m is the number of distinct design points and the discretized HRD can be obtained for any specific $m \leq n$.

Uniform Design The uniform design, denoted ξ_U is simply when the sample size n is evenly distributed over the entire design space.

Clustered Design (Cl) The clustered design is the newly proposed robust design, by Wiens (2019), which we wish to investigate. ξ_{Cl} consists of clusters distributed around the support points of a CDD. In the case of fitting a simple-linear model, the clusters would fall on the intervals $[-0.5, -0.5 + \frac{p}{2}]$ and $[0.5 - \frac{p}{2}, 0.5]$, resulting in these two intervals constituting $100p\%$ of the design space. For example, if we choose to have all $n = 30$ points be distinct and let $p = 0.1$, covering 10% of the design space, then the design would be:

$$\xi_{Cl} = \begin{pmatrix} \pm.45 & \pm.4536 & ... & \pm.4964 & \pm.5 \\ \frac{1}{30} & \frac{1}{30} & ... & \frac{1}{30} & \frac{1}{30} \end{pmatrix}$$

The set-up of the clustered design will depend on the model being fitted since a CDD is model-dependant, so each case will be described in further detail in its corresponding section.

1.3 Efficiency

The method for comparing efficiency between designs that will be used in this paper will be D-optimality. In general, the design that has the smallest determinant of the MSE matrix is the better design. So we will select a design such that the determinant of the MSE of $\hat{\boldsymbol{\beta}}$ is minimized. Namely, we find design ξ subject to:

$$\min_{\xi} det(MSE(\hat{\boldsymbol{\beta}}))$$

The design that fits this criteria is called the "Robust D-optimal Design". To compare two designs ξ_1 and ξ_2 using D-optimality we compute relative efficiency:

$$\text{Eff}(\xi_1, \xi_2) = \frac{det(MSE(\hat{\boldsymbol{\beta}}, \xi_2))}{det(MSE(\hat{\boldsymbol{\beta}}, \xi_1)}$$

The $det(MSE(\hat{\boldsymbol{\beta}}))$ is calculated using $MSE(\hat{\boldsymbol{\beta}}) = Cov(\hat{\boldsymbol{\beta}}) + Bias(\hat{\boldsymbol{\beta}})Bias(\hat{\boldsymbol{\beta}})^T$ where $Cov(\hat{\boldsymbol{\beta}}) = \sigma^2(X^TX)^{-1}$ is the covariance matrix, $Bias(\hat{\boldsymbol{\beta}}) = E(\hat{\boldsymbol{\beta}}) - \boldsymbol{\beta}$ is the bias of $\hat{\boldsymbol{\beta}}$, and $\hat{\boldsymbol{\beta}}$ is the OLS estimator for $\boldsymbol{\beta}$. As we know, $\hat{\boldsymbol{\beta}} = (X^TX)^{-1}X^T\mathbf{y}$ where $\mathbf{y} = (y_1, y_2, ..., y_n)^T$ are the observed values of the response variable, and X is the design matrix taking $z^T(\boldsymbol{x}_i)$ as its rows.

Chapter 2

Determination of Robustness Factors

We present two factors on the robustness of the newly proposed clustered design. The first one is a value which maximizes the allowed contamination subject to a given criteria, such as the L_2 neighbourhood. We will denote k as this value, which will be described further in Section 2.1. The second is to optimize the robustness protection by determining the most optimal proportion of the design space for the support points of the clustered design to cover. We will denote p as the proportion of the design space that the clustered design covers. The process of finding the optimal p will be described in Section 2.2.

2.1 Determination of k

Huber (1975) introduced a value denoted $\nu = \frac{\sigma^2}{n\eta^2} \in (0, \infty)$ which measures the relative belief of model accuracy. Its value is required when implementing HRD. See Heo (1998), Huber (1975, 1981), Wiens (1992) for more specific details on ν, but in summary, as ν approaches 0 only the bias term is involved in determining the most robust design, and the optimal design approaches the uniform design. As ν approaches ∞ only the variance term affects the MSE resulting in the CDD; where in the case of fitting $\boldsymbol{z}^T(x) = (1, x)^T$ we have $\frac{n}{2}$ design points on each endpoint. So, if ν was very large, we would be confident in fitting the simple linear model to the data; but if ν was very small, then we would abandon the simple-linear model with confidence and resort to a higher order model to fit the data. In this simulation, we will be using an example where $\nu \approx 1$, which often presents the case where the experimenter has no previous knowledge about the model accuracy.

We will present the design process by fitting Model 1 once a test for Model 2 is rejected. Other models will have a similar process. Therefore, we need to determine the approximate, maximum true value of β_2 such that we accept the hypothesis $H_0 : \beta_2 = 0$. We want to test the efficiency of both models at this

point since the outcome of the hypothesis test is most uncertain. We will denote $k = max(\beta_2)$ such that $H_0 : \beta_2 = 0$ is accepted from here on (replacing β_2 with a different parameter for other models).

We will set all other parameter values to be $\beta_0 = \beta_1 = 1$ and test the hypothesis for significance of β_2. Recall that the hypothesis test is to reject $H_0 : \beta_2 = 0$ with 95% certainty if

$$|\hat{\beta}_2| > 1.96\sqrt{Var\hat{\beta}_2}$$

We will estimate k when ξ_U is adopted, by randomly generating data according to this design of size $n = 30$ with each true value of β_2 and performing the hypothesis test N times, then lowering β_2 until we accept H_0 more than 95% ($0.95N$) of trials. Here, $N = 100$ hypothesis tests was used. We can repeat this process for whichever values of σ^2 that we wish. The process will also be averaged over 10 estimations of each k so that the estimate will be more stable. The results for these values of k for each σ^2 selected are shown in Table 2.1.

Similarly, we can find k values for Model 2 (when we are unsure if a cubic term is necessary in the fitted model) using the same process. Here β_0, β_1 and β_2 are all set equal to 1, and the hypothesis test is $|\hat{\beta}_3| > 1.96\sqrt{Var\hat{\beta}_3}$. Fedorov (1972) has $\xi^{(4)}_{CDD}$ written for design space $[-1, 1]$ which can be transformed for $[-0.5, 0.5]$ by dividing by 2, resulting in:

$$\xi^{(4)}_{CDD} = \begin{pmatrix} -0.5 & -0.2235 & 0.2235 & 0.5 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

In evaluation of Model 3: $\boldsymbol{z}^T(\boldsymbol{x}) = (1, x_1, x_2)^T$, where we are unsure of significance in just the interaction term β_{12} we can obtain values of k using the MLR uniform design. Increased n to 36 for these two models since 36 is divisible by 3 and 4 and simplifies distributing design points. Here β_0, β_1 and β_2 are all set equal to 1, and the hypothesis test is $|\hat{\beta_{12}}| > 1.96\sqrt{Var\hat{\beta_{12}}}$. The values of k for Model 2 and 3 are shown in Tables 2.2 and 2.3 respectively.

2.2 Determination of p

Clearly the choice of $p =$ (% of design space represented by design points) such that we achieve D-optimality is non-trivial. To find the best p we estimate the determinant of the MSE matrix for each $p \in (0, 1)$, increasing by increments of 0.01. In order to reduce the error in the estimates of optimal p, a simple non-parametric curve (estimated using a kernel density estimator with bandwidth 0.15) was fitted, from which the estimated D-optimal p was taken. An example of the graphs generated for Model 1, $\sigma^2 = 0.1$ is shown in Fig. 2.2.

We know that when p approaches 0, the clustered design approaches the classical design, and similarly, as p approaches 1, the clustered design resembles

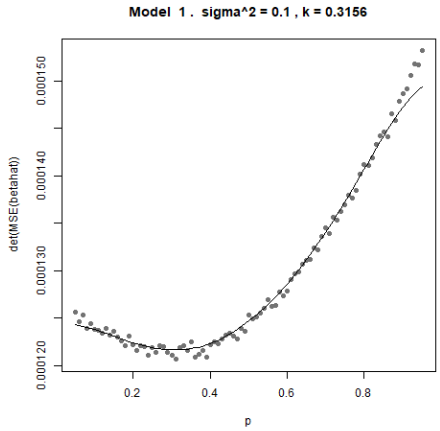


Figure 2.1: Graph of $p \in (0, 1)$.

Figure 2.2: Graph of p versus $\det(\mathrm{MSE}(\hat{\boldsymbol{\beta}}))$ for Model 1, $\sigma^2 = 0.1$. The black line shows the estimated function of p.

the uniform design, therefore we will search for the optimal p within boundary $[0.05, 0.95]$ to avoid either of these two cases. The resulting values of p which were D-optimal, where $p \in [0.05, 0.95]$ are listed in Tables 2.1, 2.2 and 2.3.

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.0040	0.0654	0.05
0.05	0.0200	0.1856	0.14
0.10	0.0400	0.3156	0.31
0.20	0.0800	0.5054	0.43
0.50	0.2000	0.8636	0.51

Table 2.1: Estimated values of k and p when fitting Model 1 with n =30

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.0033	0.3624	0.05
0.05	0.0167	0.9644	0.05
0.10	0.0333	1.5002	0.05
0.20	0.0667	2.1926	0.05
0.50	0.1667	2.8166	0.05

Table 2.2: Estimated values of k and p when fitting Model 2 with n =36

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.0033	0.0264	0.05
0.05	0.0167	0.0948	0.05
0.10	0.0333	0.1716	0.05
0.20	0.0667	0.2660	0.05
0.50	0.1667	0.4648	0.05

Table 2.3: Estimated values of k and p when fitting Model 3 with $n = 36$.

Chapter 3

Comparison

In this chapter, we will compare the efficiency of a few commonly used designs such as the CDD, HRD (Huber 1975) and uniform design.

For each model being tested, the efficiency is compared as described in Section 1.3. For sample size n, the observation vector $\boldsymbol{y}$ is generated using the true model with k being the true value of the parameter of uncertainty. This was done $N = 50000$ times and in each iteration an estimate of $\hat{\boldsymbol{\beta}}$ was obtained, from which we can estimate the $\det(\text{MSE}(\hat{\boldsymbol{\beta}}))$ using the equations defined in Section 1.3.

3.1 Simple-Linear Regression: $\boldsymbol{z}^T(x) = (1, x)^T$

Here we will be fitting Model 1, namely $Y^* = f^*(x) + \epsilon = \beta_0 + \beta_1 x + \epsilon$, to the observed data, but the true model might be quadratic (i.e. $g(x) = \beta_2 x^2$). The CDD will have the form below for this assumed simple-linear model:

$$\xi_{CDD}^{(2)} = \begin{pmatrix} -0.5 & 0.5 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

A discretized HRD will be obtained from Eq. 1.1 with $m = n = 30$ points. Huber (1975) constructed $d(x)$ for situation where design space is $[-0.5, 0.5]$ to be $d(x) = 5.12x^2 + 0.573$ when $\nu = 1$. With this information, we can discretize $d(x)$ for assumed simple-linear model with $m = n = 30$, and denote it ξ_H.

The uniform design has all $n = 30$ design points evenly spaced over the design space.

The clustered design here will consist of two clusters of design points nearby the two end points, where the percentage of the design space covered is p as found in Section 2.2 and listed in Table 2.1. For example, if $p = 0.05$ then the design will be:

$$\xi_{CL} = \begin{pmatrix} \pm.475 & \pm.477 & \dots & \pm.498 & \pm.5 \\ \frac{1}{30} & \frac{1}{30} & \cdots & \frac{1}{30} & \frac{1}{30} \end{pmatrix}$$

Here k would be the $max(\beta_2)$ such that we accept hypothesis $H_0 : \beta_2 = 0$. The results for each value of σ^2 are shown in Table 3.1 using values of k and p generated as in Section 2. The simulated bias and variance of each estimated parameter is listed in Table 3.1a and the relative efficiency of the clustered design compared to the others is shown in Table 3.1b.

Table 3.1: Model 1 Design Comparison from k_U

	$Bias(\hat{\beta}_0)$	$Bias(\hat{\beta}_1)$	$Var(\hat{\beta}_0)$	$Var(\hat{\beta}_1)$	det(MSE($\hat{\boldsymbol{\beta}}$))
	$\sigma^2 = 0.01$, $k = 0.0654$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.01643	-0.00017	0.00033	0.00133	8.0e-07
$\xi_H^{(30)}$	0.00719	-0.00034	0.00033	0.00300	1.1e-06
ξ_{Cl}	0.01560	-0.00011	0.00034	0.00141	8.1e-07
ξ_U	0.00589	-0.00024	0.00033	0.00374	1.4e-06
	$\sigma^2 = 0.05$, $k = 0.1856$, $p = 0.14$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.04683	6.0e-05	0.00168	0.00669	2.6e-05
$\xi_H^{(30)}$	0.02089	0.00017	0.00167	0.01500	3.1e-05
ξ_{Cl}	0.04032	-0.00021	0.00167	0.00770	2.5e-05
ξ_U	0.01624	0.00047	0.00166	0.01879	3.6e-05
	$\sigma^2 = 0.1$, $k = 0.3156$, $p = 0.31$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.07875	-0.00016	0.00335	0.01335	0.00013
$\xi_H^{(30)}$	0.03513	-4.4e-05	0.00334	0.02953	0.00014
ξ_{Cl}	0.05726	2.1e-05	0.00330	0.01847	0.00012
ξ_U	0.02822	0.00068	0.00336	0.03763	0.00015
	$\sigma^2 = 0.2$, $k = 0.5054$, $p = 0.43$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.12668	9.0e-04	0.00667	0.02667	0.00061
$\xi_H^{(30)}$	0.05658	0.00016	0.00668	0.05977	0.00059
ξ_{Cl}	0.07994	0.00100	0.00665	0.04193	0.00055
ξ_U	0.04491	-0.00031	0.00673	0.07473	0.00065
	$\sigma^2 = 0.5$, $k = 0.8636$, $p = 0.51$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.21708	0.00065	0.01673	0.06690	0.00425
$\xi_H^{(30)}$	0.09728	-0.00143	0.01653	0.14770	0.00390
ξ_{Cl}	0.12578	0.00019	0.01668	0.11580	0.00374
ξ_U	0.07747	0.00167	0.01670	0.18830	0.00424

(a) Table of Bias, Variance and D-optimality criteria.

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(2)}$)	Eff(ξ_{Cl},$\xi_H^{(30)}$)	Eff(ξ_{Cl},ξ_U)
0.01	0.0654	0.05	0.9948	1.4200	1.7033
0.05	0.1856	0.14	1.0163	1.2384	1.4263
0.10	0.3156	0.31	1.0429	1.1173	1.2676
0.20	0.5054	0.43	1.1027	1.0714	1.1830
0.50	0.8636	0.51	1.1385	1.0430	1.1354

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

3.2 Quadratic Regression: $\boldsymbol{z}^T(x) = (1, x, x^2)^T$

Here we want to fit the quadratic model $Y^* = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$ to the data, however we are uncertain whether there might be a significant cubic term (i.e. $g(x) = \beta_3 x^3$). Sample size is set to $n = 36$ for simplicity to analyze Model 2 since 36 is divisible by 3 and 4. The classical design when fitting a quadratic model is known to be:

$$\xi_{CDD}^{(3)} = \begin{pmatrix} -0.5 & 0 & 0.5 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

HRD can be implemented using $m = n = 30$ points once again. The density function $d(x)$ is found in Heo (1998) Table 4.2 for $\nu = 1$ to be $d(x) = 35.0934\{(x^2 - 0.1487^2)(x^2 - 0.1489^2) + 0.0192\}^+$ for the quadratic model.

Uniform design is $n = 36$ design points evenly spaced over the design space $[-0.5, 0.5]$.

Clustered design for this scenario will consist of three regions of size $p/3$ where two are located nearby the end points and the third is centered over 0. Design points are evenly spaced in this region. For example, if $p = 0.05$ we get the design:

$$\xi_{Cl} = \begin{pmatrix} \pm 0.5 & \pm 0.498 & ... & \pm 0.485 & \pm 0.483 & \pm 0.008 & \pm 0.006 & ... & \pm 0.001 \\ \frac{1}{36} & \frac{1}{36} & ... & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & ... & \frac{1}{36} \end{pmatrix}$$

Here k would be the $max(\beta_3)$ such that we accept hypothesis $H_0 : \beta_3 = 0$. The results for each value of σ^2 are shown in Table 3.2 using values of k and p generated as in Section 2. The simulated bias and variance of each estimated parameter is listed in Table 3.2a and the relative efficiency of the clustered design compared to the others is shown in Table 3.2b.

Table 3.2: Model 2 Design Comparison from k_U

	$B(\hat{\beta}_0)$	$B(\hat{\beta}_1)$	$B(\hat{\beta}_2)$	$V(\hat{\beta}_0)$	$V(\hat{\beta}_1)$	$V(\hat{\beta}_2)$	det(MSE)
	$\sigma^2 = 0.01$, $k = 0.3624$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(3)}$	6.5e-05	0.09045	0.00052	0.00082	0.00166	0.01983	5.5e-08
$\xi_H^{(36)}$	0.00016	0.06314	-4.0e-04	0.00084	0.00239	0.04112	7.2e-08
ξ_{Cl}	-2.2e-05	0.08751	0.00034	0.00083	0.00173	0.02170	5.6e-08
ξ_U	9.9e-05	0.05737	-0.00163	0.00062	0.00316	0.04499	8.0e-08
	$\sigma^2 = 0.05$, $k = 0.9644$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(3)}$	0.00016	0.24099	-0.00117	0.00414	0.00833	0.09968	9.2e-06
$\xi_H^{(36)}$	0.00017	0.16903	-0.00131	0.00415	0.01201	0.20483	1.1e-05
ξ_{Cl}	-0.00028	0.23420	0.00210	0.00413	0.00863	0.10681	9.4e-06
ξ_U	0.00036	0.15255	-0.00182	0.00311	0.01596	0.22219	1.2e-05
	$\sigma^2 = 0.1$, $k = 1.5002$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(3)}$	-0.00038	0.37511	0.00215	0.00838	0.01673	0.20118	8.7e-05
$\xi_H^{(36)}$	-0.00013	0.26231	-0.00265	0.00833	0.02386	0.41010	1.0e-04
ξ_{Cl}	-0.00054	0.36222	0.00288	0.00826	0.01732	0.21070	8.8e-05
ξ_U	0.00014	0.23825	-0.00207	0.00633	0.03165	0.45269	0.00011
	$\sigma^2 = 0.2$, $k = 2.1926$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(3)}$	-0.00055	0.54852	0.00433	0.01644	0.03340	0.39766	0.00074
$\xi_H^{(36)}$	0.00056	0.38373	-0.00712	0.01665	0.04752	0.81258	0.00088
ξ_{Cl}	3.1e-05	0.53191	-0.00250	0.01662	0.03440	0.42513	0.00075
ξ_U	1.4e-05	0.34784	-0.00120	0.01277	0.06341	0.90342	0.00092
	$\sigma^2 = 0.5$, $k = 2.8166$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(3)}$	-0.00073	0.70457	0.00381	0.04177	0.08386	1.00220	0.00805
$\xi_H^{(36)}$	-0.00119	0.49480	0.00762	0.04135	0.12070	2.01859	0.01031
ξ_{Cl}	-0.00046	0.68269	0.00101	0.04165	0.08652	1.06751	0.00819
ξ_U	-6.4e-05	0.44938	-0.00752	0.03105	0.15652	2.23722	0.01120

(a) Table of Bias, Variance and D-optimality criteria. B($\hat{\beta}$) represents bias of the estimator, V($\hat{\beta}$) represents variance of the estimator, and det(MSE) represents det(MSE($\hat{\boldsymbol{\beta}}$)).

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(3)}$)	Eff(ξ_{Cl},$\xi_H^{(36)}$)	Eff(ξ_{Cl},ξ_U)
0.01	0.3624	0.05	0.9828	1.2967	1.4421
0.05	0.9644	0.05	0.9797	1.2182	1.2913
0.10	1.5002	0.05	0.9926	1.1917	1.2487
0.20	2.1926	0.05	0.9859	1.1726	1.2174
0.50	2.8166	0.05	0.9829	1.2587	1.3670

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

3.3 Multiple-Linear Regression with two factors: $\boldsymbol{z}^T(\boldsymbol{x}) = (1, x_1, x_2)^T$

Here we want to fit the model $Y^* = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$, however we are uncertain whether there is a significant interaction term (i.e. $g(\boldsymbol{x}) = \beta_{12} x_1 x_2$). The CDD for a bivariate, square design space has equal weight on each of the four corners of the design space. So for design space $[-0.5, 0.5] \times [-0.5, 0.5]$, we have:

$$\xi_{CDD}^{(2x2)} = \begin{pmatrix} (0.5, 0.5) & (0.5, -0.5) & (-0.5, 0.5) & (-0.5, -0.5) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

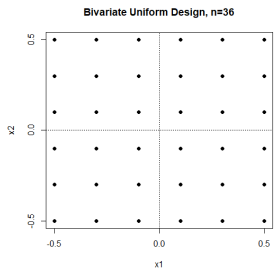
HRD is generated similarly as in Eq. 1.1 except instead we will solve for x_{j1} and x_{j2} such that

$$\int_{-0.5}^{x_{j2}} \int_{-0.5}^{x_{j1}} d(x_1, x_2) dx_1 x_2 = \frac{j - 0.5}{m}$$

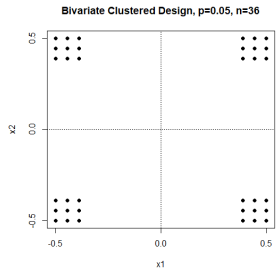
Heo (1998) found the density function for $\nu = 1.2758$ (the closest result to $\nu = 1$) to be $d(x_1, x_2) = 181.02\{(x_1^2 - 0.2333^2 + 0.044) + (x_2^2 - 0.2333^2 + 0.044)\}^+$. Since there are infinitely many solutions to this equation, we will also restrict to a symmetric design, so each design point must fall on the line $x_1 = x_2$ or $x_1 = -x_2$. With this we can construct the Huber's design with $m = n = 36$ for a bivariate study.

The uniform design consists of $n = 36$ design points evenly spaced across the entire design space $[-0.5, 0.5] \times [-0.5, 0.5]$. A figure of the design space is shown in Fig. 3.1a.

The clustered design for this scenario will consist of grids of design points evenly spaced at each of the four corners of the design space. An example of what the design looks like for $p = 0.05$ is shown in Fig. 3.1b. A square-shaped design was selected primarily for simplicity, but changing the shape of the layout for design points in each corner should not have too significant of an effect on the efficiency of the design.



(a) Bivariate Uniform Design



(b) Clustered Design with $p = 0.05$.

Here k would be the $max(\beta_{12})$ such that we accept hypothesis $H_0 : \beta_{12} = 0$. The results for each value of σ^2 are shown in Table 3.3 using values of k and p

generated as in Section 2. The simulated bias and variance of each estimated parameter when using k as the true value of β_{12}, and p to generate the clustered design is listed in Table 3.3a and the relative efficiency of the clustered design compared to the others is shown in Table 3.3b.

Table 3.3: Model 3 Design Comparison from k_U

	$B(\hat{\beta}_0)$	$B(\hat{\beta}_1)$	$B(\hat{\beta}_2)$	$V(\hat{\beta}_0)$	$V(\hat{\beta}_1)$	$V(\hat{\beta}_2)$	det(MSE)
	$\sigma^2 = 0.01$, $k = 0.0264$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	-0.00014	7.5e-05	0.00032	0.00028	0.00112	0.00111	3.4e-10
$\xi_H^{(36)}$	9.6e-05	0.00013	9.2e-05	0.00028	0.00139	0.00141	5.4e-10
ξ_{Cl}	1.6e-05	0.00017	0.00035	0.00028	0.00140	0.00139	5.4e-10
ξ_U	3.3e-05	-1.8e-06	-8.4e-05	0.00028	0.00236	0.00239	1.6e-09
	$\sigma^2 = 0.05$, $k = 0.0948$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	1.0e-04	-4.3e-05	-6.0e-04	0.00139	0.00558	0.00553	4.3e-08
$\xi_H^{(36)}$	1.6e-06	0.00058	0.00048	0.00139	0.00697	0.00706	6.8e-08
ξ_{Cl}	3.5e-06	0.00041	-8.6e-05	0.00139	0.00696	0.00699	6.7e-08
ξ_U	8.9e-05	-7.8e-05	0.00021	0.00139	0.01200	0.01204	2.0e-07
	$\sigma^2 = 0.1$, $k = 0.1716$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.00042	0.00048	0.00059	0.00277	0.01119	0.01113	3.4e-07
$\xi_H^{(36)}$	-0.00043	-0.00012	0.00072	0.00279	0.01410	0.01405	5.4e-07
ξ_{Cl}	-2.4e-05	0.00055	5.0e-04	0.00282	0.01379	0.01392	5.4e-07
ξ_U	0.00024	-0.00066	-0.00024	0.00276	0.02393	0.02380	1.6e-06
	$\sigma^2 = 0.2$, $k = 0.266$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	-0.00031	0.00019	-0.00057	0.00555	0.02234	0.02230	2.7e-06
$\xi_H^{(36)}$	0.00029	-0.00023	0.00052	0.00555	0.02786	0.02802	4.3e-06
ξ_{Cl}	-6.5e-05	0.00104	0.00023	0.00556	0.02794	0.02769	4.3e-06
ξ_U	0.00036	-0.00014	-0.00120	0.00555	0.04744	0.04757	1.3e-05
	$\sigma^2 = 0.5$, $k = 0.4648$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	-0.00057	0.00055	0.00034	0.01376	0.05588	0.05538	4.3e-05
$\xi_H^{(36)}$	0.00011	0.00039	-0.00119	0.01393	0.06935	0.06990	6.8e-05
ξ_{Cl}	-7.3e-05	0.00062	-0.00023	0.01388	0.07028	0.06966	6.7e-05
ξ_U	-0.00038	-0.00089	0.00077	0.01391	0.11908	0.11915	2.0e-04

(a) Table of Bias, Variance and D-optimality criteria. B($\hat{\beta}$) represents bias of the estimator, V($\hat{\beta}$) represents variance of the estimator, and det(MSE) represents det(MSE($\hat{\boldsymbol{\beta}}$)).

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(2x2)}$)	Eff(ξ_{Cl},$\xi_H^{(36)}$)	Eff(ξ_{Cl},ξ_U)
0.01	0.0264	0.05	0.6356	1.0032	2.9181
0.05	0.0948	0.05	0.6356	1.0034	2.9184
0.10	0.1716	0.05	0.6356	1.0034	2.9184
0.20	0.2660	0.05	0.6356	1.0033	2.9185
0.50	0.4648	0.05	0.6356	1.0033	2.9185

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

3.4 Discussion

Model 1: Simple-Linear Model As we can see from the results, the clustered design is highly comparable with the CDD and far outperforms HRD and uniform for D-optimality. The det(MSE) for clustered design is very similar to that of CDD for most σ^2 tested. The clustered design has lower bias for β_0 across all σ^2 but generally no improvement for the bias of the slope β_1. We also see that the variance of the estimators changes very little between clustered and CDD. HRD appears to mainly reduce bias in β_0 even more so than the clustered design but at the cost of a bigger increase in variance.

If the parameter of primary concern is β_0 then the clustered design may be beneficial to implement due to the reasonable decrease in bias and just a small increase in variance. If the primary parameter of concern is the slope β_1 then the design of choice is left to the researcher, since there is no clear design that consistently has the lowest bias of β_1.

Model 2: Quadratic Model Once again, for the quadratic model we see a similar effect. CDD and clustered both far outperform HRD and uniform in terms of D-optimality. The clustered design is nearly identical in det(MSE) and variance of the estimators to CDD. Overall, the clustered design seems to improve the bias of β_0 and β_1, compared with the CDD, but does not help with β_2. HRD reduces bias greatly in β_1 compared with the CDD but not for β_0 and β_2. But HRD requires a much greater sacrifice in increased variance, and therefore efficiency.

If the parameter of interest is β_0 and/or β_1 then the clustered design may again be beneficial to implement, since it reduces the bias of those two parameters compared with the CDD while only increasing variance by a small amount. If the parameter of interest is only β_1 then HRD may be ideal to the researcher. But if the parameter of interest is β_2 then there is no clear best design to choose.

Model 3: MLR Model In the bivariate MLR Model, the clustered design doesn't perform as well as the univariate cases. The CDD has a much better efficiency than clustered, while clustered design performs more similarly to HRD. Both the clustered design and HRD have varying effects on the bias of any of the three parameters compared with the CDD. In the MLR setting, it would seem that the best option is still the CDD where design points are evenly distributed over the four corners.

Chapter 4

More Complex Contamination Functions

In this chapter, we will further consider the case with some more complicated contamination functions with, for example, several terms missing in the fitted model. This can often happen when the experimenter is fitting a lower order polynomial when the true model has a much higher order. For Model 1, we could have a contamination function that looks something like $g(x) = \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$. In this case how does the clustered design perform compared with other designs?

Here k can't be found in the same way as before since we have three parameters, which would each have to have their own hypothesis test. Instead we will find k based on the L_2 contamination space. Let the fitted model be $Y^* = f^*(x) + \epsilon = \beta_0^* + \beta_1^* x + \epsilon$, but the true model is $Y = f(x) + \epsilon = \beta_0 + \beta_1 x + k(\beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4) + \epsilon$. We can specify β_2, β_3 and β_4 and then find the maximum value of k such that

$$\int_S [f(\boldsymbol{x}) - f^*(\boldsymbol{x})]^2 dx \leq \eta^2 \tag{4.1}$$

where $\eta^2 = \frac{\sigma^2}{n\nu}$ and the integral is over the entire design space. But since we are concerned with when $\nu = 1$, the boundary becomes $\eta^2 = \frac{\sigma^2}{n}$, which can be easily calculated for each value of σ^2 that we test. The β^*'s will be found such that we minimize the left side of Eq. 4.1 over all β^*'s.

The simulation and comparison of designs is run in the exact same process as in the previous section, but with the new k and p values.

4.1 Simple-Linear Regression: $\boldsymbol{z}^T(x) = (1, x)^T$

As in the example above, we will find a sufficient k for comparing the designs by minimizing Eq. 4.1. Parameters β_2, β_3 and β_4 are set equal to 1. Taking

the derivative with respect to each β^*, integrating and setting both equations equal to zero, and then solving the system of equations we obtain:

$$\beta_0^* = 1 + k\frac{23}{240} \qquad \beta_1^* = 1 + k\frac{3}{20}$$

Then we can decrease k until we find the $max(k)$ such that Eq. 4.1 is satisfied. The resulting values of k are shown in Table 4.1. Values of p are calculated in the same way as the last section but with the new k values. A comparison of simulated biases and variances of estimated parameters for each of the four different designs is shown in Table 4.4a. Table 4.4b compares the relative efficiency of each design with the clustered design according to D-optimality.

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.004	0.0167	0.05
0.05	0.020	0.0374	0.05
0.10	0.040	0.0528	0.05
0.20	0.080	0.0747	0.05
0.50	0.200	0.1182	0.05

Table 4.1: Estimated values of k and p when fitting Model 1 with n =30

4.2 Quadratic Regression: $\boldsymbol{z}^T(x) = (1, x, x^2)^T$

In this scenario, the experimenter fits a quadratic model to the data, however the true model is much more complicated than just quadratic. Here k is found with fitted model $Y^* = \beta_0^* + \beta_1^* x + \beta_2^* x^2 + \epsilon$ and true model is $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + k(x^3 + x^4 + x^5) + \epsilon$. So we will find k subject to Eq. 4.1 where β^*'s are again found such that the value of the integral is minimized. Parameters β_3, β_4 and β_5 are set equal to 1. Taking the derivative with respect to each β^*, integrating and solving the system of equations yields:

$$\beta_0^* = 1 - k\frac{3}{560} \qquad \beta_1^* = 1 + k\frac{99}{560} \qquad \beta_2^* = 1 + k\frac{3}{14}$$

Then we can decrease k until we find the $max(k)$ such that the L_2 boundary condition is satisfied. The resulting values of k are shown in Table 4.2. Values of p are again calculated in the same way as the last section but with the new k values. A comparison of simulated biases and variances of estimated parameters for each of the designs is shown in Table 4.5a. Table 4.5b compares the relative efficiency of each design with the clustered design according to D-optimality.

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.0033	0.6763	0.71
0.05	0.0167	1.5124	0.71
0.10	0.0333	2.1388	0.71
0.20	0.0667	3.0247	0.71
0.50	0.1667	4.7826	0.72

Table 4.2: Estimated values of k and p when fitting Model 2 with n =36

4.3 Multiple-Linear Regression with two factors: $\boldsymbol{z}^T(\boldsymbol{x}) = (1, x_1, x_2)^T$

In this situation, the experimenter fits a first-order MLR model to the data, however the true model is much more complicated and involves all second-order terms. Here k is found with fitted model $Y^* = \beta_0^* + \beta_1^* x_1 + \beta_2^* x_2 + \epsilon$ and true model is $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + k(x_1 x_2 + x_1^2 + x_2^2) + \epsilon$. We will find k subject to Eq. 4.1 again except the integral becomes a double integral:

$$\int\int_S [f^*(\boldsymbol{x}) - f^*(\boldsymbol{x})]^2 dx_1 dx_2 \leq \eta^2$$

where β^*'s are found such that the value of the integral is minimized. Parameters β_3, β_4 and β_5 are set equal to 1. Taking the derivative with respect to each β^*, integrating, and solving the system of equations yields:

$$\beta_0^* = 1 + k\frac{1}{6} \qquad \beta_1^* = \beta_2^* = 1$$

Then we can decrease k until we find the $max(k)$ such that the L_2 boundary condition is satisfied. The resulting values of k are shown in Table 4.3. Values of p are calculated in the same way as last section but with new k values. A comparison of simulated biases and variances of estimated parameters for each of the designs is shown in Table 4.6a. Table 4.6b compares the relative efficiency of each design with the clustered design according to D-optimality.

σ^2	$12\frac{\sigma^2}{n}$	k	p
0.01	0.0033	0.1240	0.05
0.05	0.0167	0.2773	0.05
0.10	0.0333	0.3922	0.05
0.20	0.0667	0.5547	0.05
0.50	0.1667	0.8770	0.05

Table 4.3: Estimated values of k and p when fitting Model 3 with n =36

Table 4.4: Model 1 Design Comparison where contamination function is more complex.

	$Bias(\hat{\beta}_0)$	$Bias(\hat{\beta}_1)$	$Var(\hat{\beta}_0)$	$Var(\hat{\beta}_1)$	det(MSE($\hat{\boldsymbol{\beta}}$))
	$\sigma^2 = 0.01$, $k = 0.0167$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.00521	0.00434	0.00033	0.00134	4.9e-07
$\xi_H^{(3)}$	0.00186	0.00310	0.00033	0.00316	1.1e-06
$\xi_H^{(30)}$	0.00218	0.00284	0.00033	0.00300	1e-06
ξ_{Cl}	0.00500	0.00394	0.00034	0.00141	5.1e-07
ξ_U	0.00166	0.00269	0.00033	0.00372	1.3e-06
	$\sigma^2 = 0.05$, $k = 0.0374$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.01168	0.01032	0.00166	0.00673	1.2e-05
$\xi_H^{(3)}$	0.00473	0.00702	0.00167	0.01594	2.7e-05
$\xi_H^{(30)}$	0.00480	0.00665	0.00168	0.01496	2.5e-05
ξ_{Cl}	0.01089	0.00844	0.00168	0.00704	1.3e-05
ξ_U	0.00393	0.00608	0.00167	0.01879	3.2e-05
	$\sigma^2 = 0.1$, $k = 0.0528$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.01641	0.01346	0.00333	0.01333	4.9e-05
$\xi_H^{(3)}$	0.00618	0.00813	0.00337	0.03182	0.00011
$\xi_H^{(30)}$	0.00649	0.00764	0.00330	0.03002	1.0e-04
ξ_{Cl}	0.01558	0.01307	0.00331	0.01406	5.1e-05
ξ_U	0.00583	0.00854	0.00333	0.03750	0.00013
	$\sigma^2 = 0.2$, $k = 0.0747$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.02339	0.01883	0.00671	0.02671	0.00019
$\xi_H^{(3)}$	0.00947	0.01083	0.00668	0.06313	0.00043
$\xi_H^{(30)}$	0.00971	0.01315	0.00668	0.05961	4.0e-04
ξ_{Cl}	0.02247	0.01764	0.00670	0.02775	2.0e-04
ξ_U	0.00784	0.01036	0.00671	0.07511	5.0e-04
	$\sigma^2 = 0.5$, $k = 0.1182$, $p = 0.05$, $n = 30$				
$\xi_{CDD}^{(2)}$	0.03670	0.02991	0.01666	0.06623	0.00122
$\xi_H^{(3)}$	0.01383	0.01678	0.01676	0.15984	0.00269
$\xi_H^{(30)}$	0.01523	0.01841	0.01657	0.15029	0.00253
ξ_{Cl}	0.03500	0.02980	0.01662	0.07030	0.00127
ξ_U	0.01271	0.01619	0.01645	0.18693	0.00315

(a) Table of Bias, Variance and D-optimality criteria.

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(2)}$)	Eff(ξ_{Cl},$\xi_H^{(3)}$)	Eff(ξ_{Cl},ξ_H^{30})	Eff(ξ_{Cl},ξ_U)
0.01	0.0167	0.05	0.9589	2.1214	1.9909	2.4819
0.05	0.0374	0.05	0.9653	2.1374	1.9997	2.4955
0.10	0.0528	0.05	0.9591	2.1241	1.9887	2.4893
0.20	0.0747	0.05	0.9583	2.1242	1.9901	2.4815
0.50	0.1182	0.05	0.9580	2.1212	1.9897	2.4841

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

Table 4.5: Model 2 Design Comparison where contamination function is more complex.

	$B(\hat{\beta}_0)$	$B(\hat{\beta}_1)$	$B(\hat{\beta}_2)$	$V(\hat{\beta}_0)$	$V(\hat{\beta}_1)$	$V(\hat{\beta}_2)$	det(MSE)
	$\sigma^2 = 0.01$, $k = 0.6763$, $p = 0.71$, $n = 36$						
$\xi_{CDD}^{(3)}$	-2.7e-05	0.21134	0.16913	8.3e-06	1.7e-05	2.0e-04	3.0e-07
$\xi_H^{(4)}$	-0.00437	0.13669	0.15351	8.4e-06	2.4e-05	0.00043	2.9e-07
$\xi_H^{(36)}$	-0.00470	0.14139	0.15863	8.3e-06	2.4e-05	4.0e-04	2.9e-07
ξ_{Cl}	-0.00357	0.13659	0.14958	7.0e-06	2.7e-05	0.00041	2.7e-07
ξ_U	-0.00401	0.12728	0.15260	6.3e-06	3.2e-05	0.00045	2.7e-07
	$\sigma^2 = 0.05$, $k = 1.5124$, $p = 0.71$, $n = 36$						
$\xi_{CDD}^{(3)}$	-6.7e-07	0.47264	0.37815	0.00021	0.00042	0.00499	3.7e-05
$\xi_H^{(4)}$	-0.00975	0.30551	0.34329	0.00021	0.00061	0.01087	3.6e-05
$\xi_H^{(36)}$	-0.01044	0.31617	0.35442	0.00021	6.0e-04	0.01009	3.6e-05
ξ_{Cl}	-0.00803	0.30521	0.33490	0.00018	0.00067	0.01021	3.4e-05
ξ_U	-0.00896	0.28494	0.34184	0.00015	0.00079	0.01115	3.4e-05
	$\sigma^2 = 0.1$, $k = 2.1388$, $p = 0.71$, $n = 36$						
$\xi_{CDD}^{(3)}$	0.00027	0.66829	0.53408	0.00084	0.00167	0.02017	3.0e-04
$\xi_H^{(4)}$	-0.01398	0.43231	0.48624	0.00083	0.00244	0.04304	0.00029
$\xi_H^{(36)}$	-0.01484	0.44735	0.50200	0.00083	0.00238	0.04096	0.00029
ξ_{Cl}	-0.01140	0.43157	0.47484	0.00071	0.00271	0.04054	0.00027
ξ_U	-0.01270	0.40258	0.48389	0.00063	0.00319	0.04505	0.00027
	$\sigma^2 = 0.2$, $k = 3.0247$, $p = 0.71$, $n = 36$						
$\xi_{CDD}^{(3)}$	0.00011	0.94494	0.75477	0.00333	0.00663	0.07979	0.00238
$\xi_H^{(4)}$	-0.01927	0.61086	0.68472	0.00334	0.00989	0.17274	0.00231
$\xi_H^{(36)}$	-0.02105	0.63191	0.70959	0.00333	0.00946	0.16218	0.00231
ξ_{Cl}	-0.01640	0.61055	0.67345	0.00279	0.01104	0.16008	0.00217
ξ_U	-0.01804	0.56987	0.68334	0.00249	0.01265	0.17887	0.00220
	$\sigma^2 = 0.5$, $k = 4.7826$, $p = 0.72$, $n = 36$						
$\xi_{CDD}^{(3)}$	-0.00131	1.49510	1.20149	0.02081	0.04157	0.50251	0.03717
$\xi_H^{(4)}$	-0.03051	0.96709	1.08649	0.02093	0.06101	1.07634	0.03612
$\xi_H^{(36)}$	-0.03305	0.99839	1.11717	0.02080	0.05991	1.01712	0.03596
ξ_{Cl}	-0.02517	0.96361	1.05849	0.01746	0.06881	1.01119	0.03403
ξ_U	-0.02887	0.90045	1.08564	0.01562	0.07807	1.11397	0.03432

(a) Table of Bias, Variance and D-optimality criteria.

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(3)}$)	Eff(ξ_{Cl},$\xi_H^{(4)}$)	Eff(ξ_{Cl},$\xi_H^{(36)}$)	Eff(ξ_{Cl},ξ_U)
0.01	0.6763	0.71	1.0942	1.0625	1.0627	1.0095
0.05	1.5124	0.71	1.0956	1.0629	1.0639	1.0126
0.10	2.1388	0.71	1.0951	1.0634	1.0646	1.0110
0.20	3.0247	0.71	1.0934	1.0611	1.0616	1.0111
0.50	4.7826	0.72	1.0924	1.0616	1.0567	1.0085

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

Table 4.6: Model 3 Design Comparison where contamination function is more complex.

	$B(\hat{\beta}_0)$	$B(\hat{\beta}_1)$	$B(\hat{\beta}_2)$	$V(\hat{\beta}_0)$	$V(\hat{\beta}_1)$	$V(\hat{\beta}_2)$	det(MSE)
	$\sigma^2 = 0.01$, $k = 0.124$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.06201	1.7e-06	1.2e-05	2.8e-06	1.1e-05	1.1e-05	5.1e-09
$\xi_H^{(4)}$	0.04864	2.5e-05	-1.1e-05	2.8e-06	1.4e-05	1.4e-05	5.3e-09
$\xi_H^{(36)}$	0.04934	1.0e-05	-3.0e-05	2.8e-06	1.4e-05	1.4e-05	5.3e-09
ξ_{Cl}	0.04942	1.1e-05	1.6e-05	2.8e-06	1.4e-05	1.4e-05	5.3e-09
ξ_U	0.02894	-1.5e-05	-7.3e-06	2.8e-06	2.4e-05	2.4e-05	6.3e-09
	$\sigma^2 = 0.05$, $k = 0.2773$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.13859	-1.0e-04	-6.3e-06	6.9e-05	0.00028	0.00028	6.4e-07
$\xi_H^{(4)}$	0.10874	-0.00021	-7.8e-05	7.0e-05	0.00035	0.00035	6.6e-07
$\xi_H^{(36)}$	0.11032	-2.8e-05	-2.7e-06	6.9e-05	0.00035	0.00035	6.6e-07
ξ_{Cl}	0.11063	-1.1e-05	-9.2e-05	6.9e-05	0.00035	0.00035	6.6e-07
ξ_U	0.06471	-8.3e-05	-0.00028	6.9e-05	6.0e-04	0.00059	7.9e-07
	$\sigma^2 = 0.1$, $k = 0.3922$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.19601	3.4e-05	-0.00017	0.00028	0.00113	0.00111	5.1e-06
$\xi_H^{(4)}$	0.15374	6.0e-05	-9.8e-05	0.00028	0.00142	0.00142	5.3e-06
$\xi_H^{(36)}$	0.15615	-0.00016	-0.00011	0.00028	0.00140	0.00140	5.3e-06
ξ_{Cl}	0.15640	3.9e-05	8.3e-05	0.00028	0.00139	0.00141	5.3e-06
ξ_U	0.09149	-0.00019	-5.0e-04	0.00028	0.00238	0.00237	6.3e-06
	$\sigma^2 = 0.2$, $k = 0.5547$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.27718	0.00049	-0.00012	0.00111	0.00450	0.00444	4.1e-05
$\xi_H^{(4)}$	0.21747	0.00013	0.00038	0.00112	0.00568	0.00568	4.2e-05
$\xi_H^{(36)}$	0.22068	6.9e-05	-0.00016	0.00111	0.00554	0.00556	4.2e-05
ξ_{Cl}	0.22082	-9.2e-05	-0.00018	0.00111	0.00556	0.00560	4.2e-05
ξ_U	0.12944	0.00023	0.00026	0.00111	0.00955	0.00956	5.1e-05
	$\sigma^2 = 0.5$, $k = 0.877$, $p = 0.05$, $n = 36$						
$\xi_{CDD}^{(2x2)}$	0.43835	0.00043	-0.00021	0.00700	0.02757	0.02796	0.00064
$\xi_H^{(4)}$	0.34377	-0.00146	-0.00094	0.00697	0.03532	0.03552	0.00066
$\xi_H^{(36)}$	0.34895	-4.0e-04	-0.00052	0.00697	0.03500	0.03503	0.00066
ξ_{Cl}	0.34989	-0.00106	0.00064	0.00694	0.03508	0.03478	0.00066
ξ_U	0.20491	-0.00147	-0.00067	0.00695	0.05978	0.05925	0.00079

(a) Table of Bias, Variance and D-optimality criteria.

σ^2	k	p	Eff(ξ_{Cl},$\xi_{CDD}^{(2x2)}$)	Eff(ξ_{Cl},$\xi_H^{(4)}$)	Eff(ξ_{Cl},$\xi_H^{(36)}$)	Eff(ξ_{Cl},ξ_U)
0.01	0.1240	0.05	0.9634	1.0042	1.0002	1.1966
0.05	0.2773	0.05	0.9606	1.0018	0.9983	1.1941
0.10	0.3922	0.05	0.9613	1.0020	1.0005	1.1946
0.20	0.5547	0.05	0.9640	1.0054	1.0022	1.1988
0.50	0.8770	0.05	0.9607	1.0011	0.9985	1.1963

(b) Efficiency Comparison of det(MSE($\hat{\boldsymbol{\beta}}$)) between designs.

4.4 Discussion

Linear Model In this situation with the complex contamination function, we see that the bias in estimates of β_0 and β_1 does decrease somewhat, while the variance of β_1 increases slightly as we might expect. HRD and uniform design have much lower bias in the estimates, but at the cost of a greater increase in variance. The CDD remains the most D-optimal outperforming the other designs in efficiency with the clustered design close behind.

Therefore, the clustered design may be useful to implement, since it helps reduce bias in both β_0 and β_1, while only costing a small amount of variance in β_1. Overall, the clustered design performs better when there are more complex contamination functions that when we just have $g(x) = x^2$. So if a researcher is looking to fit a simple-linear model to some data, but is unsure about many other higher order terms, the clustered design does well.

Quadratic Model In the case of quadratic model with more complex contamination function, the clustered design appears to be slightly more D-optimal than all the other designs, though by a small margin. The clustered design does us no help in estimation of β_0, however we see a reduction in bias of β_1 and a slight reduction in bias of β_2. However the variance of both these estimates approximately doubles when using the clustered design compared to the CDD. The clustered design improves the reduction in bias of β_0 and β_2 even more than all of the other designs, but the uniform design still reduces bias of β_1 the most. The clustered design appears to be beneficial to implement in this scenario as well, particularly if estimation of β_0 is unimportant to the researcher.

Bivariate Model In the scenario of fitting Model 3, all designs are fairly similar in efficiency, with the CDD being most efficient, followed closely by HRD and clustered design. Comparing the bias for the CDD and the clustered design, we can see that the clustered design greatly reduces bias in estimation of β_0, but the effect for β_1 and β_2 seems to vary. Variance of β_0 appears to stay the same across each design. Variance of β_1 and β_2 are similar for clustered design and HRD but the CDD consistently has the lower variance as we might expect. If the parameter of interest is β_0 then the clustered design appears to be very beneficial to implement since the bias is reduced with negligible increase in variance. If both β_1 and β_2 are the parameters of interest, then likely remaining with the CDD is the best option.

Chapter 5

Application in Extrapolation

Extrapolation has many practical uses in research and data analysis. One instance where extrapolation is used is in Accelerated Life Testing (ALT). Typically, if an engineer is testing a product, they will conduct an experiment where the product is being operated under normal conditions. However, it can sometimes take a long time to get data for characteristics like the product's life-expectancy, so the engineer may wish to implement ALT instead. Here the product will be tested under extreme conditions (such as temperature, voltage, vibration, etc...) that are much higher than normal, and then extrapolate the life-data back to normal conditions. This can save the engineer a lot of time and money in the product-testing process. So, if the clustered design improves extrapolation, ALT is a subject that would benefit greatly.

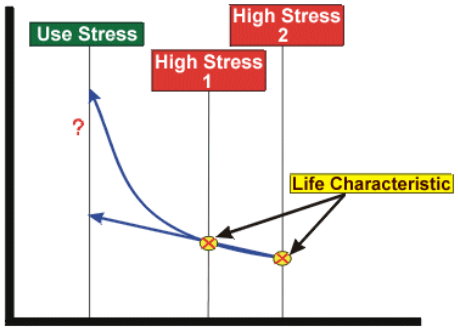


Figure 5.1: Visual concept of Accelerated Life Testing.

Since, we have no idea what the true model for the data will look like outside of the sample space, a robust design that is less sensitive to errors in the assumed model will be necessary. In this chapter we will test the robustness of

the clustered design in some extrapolation scenarios. We have the same contamination limitations here as in the L_2 neighbourhood, except we must add another boundary condition which puts a limit on the amount of bias between the endpoint 0.5 and the extrapolation point. This condition is written as:

$$\int_{0.5}^{x_0} [f(x) - f^*(x)]^2 \leq \eta^2 \tag{5.1}$$

5.1 Efficiency

Since we are focusing on accuracy of prediction on an extrapolation point rather than accuracy of parameter estimation, our MSE will have to be calculated differently. In this scenario, we can actually use the asymptotic formulas for bias, variance and MSE of $\hat{y}(x_0)$. The formulas we use to compare are: $Var(\hat{y}(x_0)) = \sigma^2 \boldsymbol{z}(x_0)(X^TX)^{-1}\boldsymbol{z}^T(x_0)$, where $\boldsymbol{z}(x_0) = (1, x_0)$ for Model 1 and $\boldsymbol{z}(x_0) = (1, x_0, x_0^2)$ for Model 2; $Bias(\hat{y}(x_0)) = \boldsymbol{z}(x_0)(X^TX)^{-1}X^TE(\boldsymbol{y}) - f(x_0)$, and $MSE(\hat{y}(x_0)) = Var(\hat{y}(x_0)) + [Bias(\hat{y}(x_0))]^2$. Since we are no longer dealing with a matrix, we are using Q-optimality as our criterion for extrapolation instead of D-optimality.

5.2 Designs used for comparison

Once again, for extrapolation we will be comparing a few different designs, including the Hoel-Levine design (Hoel and Levine 1964), Wiens-Xu design (Wiens and Xu 2008), the clustered design, and the uniform design. We will compare the efficiency of each design using two extrapolation points $x_0 = 0.75$ and $x_0 = 2.5$. Each design will be described in more detail in the following paragraphs.

CDD The CDD in the extrapolation case is taken from Hoel and Levine (1964) who found the optimal design minimizing extrapolation variance alone, denoted ξ_{HL}. The HL design when fitting Model 1 after transformation is:

$$\xi_{HL} = \begin{pmatrix} -0.5 & 0.5 \\ \frac{2x_0-1}{4x_0} & \frac{2x_0+1}{4x_0} \end{pmatrix}$$

The HL design for Model 2 after being transformed is:

$$\xi_{HL} = \begin{pmatrix} -0.5 & 0 & 0.5 \\ \frac{x_0(2x_0-1)}{8x_0^2-1} & \frac{(2x_0+1)(2x_0-1)}{8x_0^2-1} & \frac{x_0(2x_0+1)}{8x_0^2-1} \end{pmatrix}$$

Optimal Robust Design Weins and Xu (2008) constructed the D-optimal design by minimizing the determinant of the MSE of the prediction at extrapolation point, denoted ξ_{WX}. The density function of ξ_{WX} after transformation

and normalization for Model 1 found by Weins and Xu (2008) for $\nu = 1$ (for ν as described in Section 2.1) is:

$$d(x) = \frac{[1264.4148x^2 + 199.2849x - 26.7067]^+}{(22.46x+1)^2 + 0.0032(22.46x+1)^4} \text{ , and}$$

$$d(x) = \frac{[55064.7918x^2 + 2144.8033x - 594.7690]^+}{(160.76x+1)^2 + 0.000122(160.76x+1)^4}$$

for $x_0 = 0.75$ and $x_0 = 2.5$ respectively.

The density function of ξ_{WX} after transformation and normalization when fitting Model 2 is:

$$d(x) = \frac{[(1-0.142x-8.68x^2)(2.5387-3.5418x-33.8493x^2)-0.8173]^+}{(1-0.142x-8.68x^2)^2 + 0.627(1-0.142x-8.68x^2)^4} \text{ , and}$$

$$d(x) = \frac{[(1-0.0158x-8.44x^2)(3.3922-1.0237x-40.2248x^2)-0.9675]^+}{(1-0.0158x-8.44x^2)^2 + 0.949(1-0.0158x-8.44x^2)^4}$$

for $x_0 = 0.75$ and $x_0 = 2.5$ respectively.

Uniform Design The uniform design is as previously used, consisting of n design support points distributed evenly over the design space.

Clustered Design The clustered design for Model 1 will consist of design points spread out over m areas of size p/m located at each design support point of the corresponding HL design. The number of points distributed over each area however will be nw_i where the w_i's are the same weights calculated for HL design. After testing a few different ways of setting up the clustered design, this method was consistently the most optimal. See Section 5.6 for more details.

5.3 Model 1: $\boldsymbol{z}^T(x) = (1, x)^T$

Determination of k's and p Consider the case of fitting Model 1 to the data (i.e. $Y^* = \beta_0^* + \beta_1^* x + \epsilon$), however the true model is actually $Y = \beta_0 + \beta_1 x + k_2\beta_2 x^2 + k_3\beta_3 x^3 + \epsilon$. We will let all true β values be set equal to 1 and we solve for β_0^*, β_1^*, k_2 and k_3 using the following four equations:

$$\frac{\partial}{\partial \beta_i^*} \int_{-0.5}^{0.5} [f(x) - f^*(x)]^2 dx = 0, \quad i = 0, 1,$$

$$\int_{-0.5}^{0.5} [f(x) - f^*(x)]^2 dx = \eta^2, \quad \text{and}$$

$$\int_{0.5}^{x_0} [f(x) - f^*(x)]^2 dx = \eta^2.$$

The resulting values of k_2 and k_3 (and β^*'s for interest) are shown in Table 5.1 for $x_0 = 0.75$ and in Table 5.2 for $x_0 = 2.5$. The optimal p values with minimum value taken to be $p = 0.05$ are calculated in the same way as in previous sections, except using the values k_2 and k_3 in the true function found in the previous step.

σ^2	$\frac{\sigma^2}{n}$	β_0^*	β_1^*	k_2	k_3	p
0.01	0.0003	1.0198	0.9645	0.2375	-0.2367	0.05
0.05	0.0017	1.0443	0.9206	0.5310	-0.5293	0.05
0.10	0.0033	1.0626	0.8877	0.7510	-0.7485	0.05
0.20	0.0067	1.0885	0.8412	1.0621	-1.0586	0.05
0.50	0.0167	1.1399	0.7489	1.6793	-1.6738	0.05

Table 5.1: Values of k_2, k_3 and p when fitting Model 1 for extrapolation at $x_0 = 0.75$, with $n = 30$.

σ^2	$\frac{\sigma^2}{n}$	β_0^*	β_1^*	k_2	k_3	p
0.01	0.0003	1.0021	0.9982	0.0250	-0.0120	0.05
0.05	0.0017	1.0040	0.9963	0.0480	-0.0250	0.05
0.10	0.0033	1.0066	0.9943	0.0790	-0.0380	0.05
0.20	0.0067	1.0093	0.9924	0.1110	-0.0510	0.05
0.50	0.0167	1.0140	0.9874	0.1680	-0.0840	0.05

Table 5.2: Values of k_2, k_3 and p when fitting Model 1 for extrapolation at $x_0 = 2.5$, with $n = 30$.

Comparison of Designs To compare designs we can compute the asymptotic extrapolation bias, extrapolation variance and extrapolation MSE for each design using the formulas listed in Section 5.1. Table 5.3 shows a comparison of the four designs when $x_0 = 0.75$ and Table 5.5 shows the comparison when $x_0 = 2.5$.

To further compare the designs for extrapolation we will compute a couple other measures. The first is relative bias (RB), which is computed using

$$RB = (\overline{\hat{y}(x_0)} - f(x_0))/f(x_0) \times 100\%$$

Secondly, we can compute the prediction interval (PI) of the jth run given by the formula

$$PI_{(j)} = \hat{y}(x_0)_{(j)} \pm t_{\frac{\alpha}{2}}(n-q)\hat{\sigma}_{(j)}\sqrt{z^T(x_0)(X^TX)^{-1}z(x_0)+1}$$

, where $\hat{\sigma}^2_{(j)} = \sum_{i=1}^{n}(y_i - \hat{y}_{(j)})^2/(n-q)$ and q is the number of fitted parameters. We take $\alpha = 0.05$ here. The PI coverage percentage is then the number of prediction intervals out of all $N = 50,000$ trials that contain the true value

$f(x_0)$ and divide by N (ie. the percentage of prediction intervals which contain the true value $f(x_0)$). We will also compute the average length of all N 95% prediction intervals.

Table 5.4 shows the above values of interest for Model 1, when $x_0 = 0.75$. Table 5.6 shows the values of interest for each of the four designs for Model 1 when $x_0 = 2.5$.

	$Bias(\hat{y}(x_0))$	$Var(\hat{y}(x_0))$	$MSE(\hat{y}(x_0))$	Eff(ξ_{Cl},ξ)
	$\sigma^2 = 0.01$, $k_2 = 0.2327$, $k_3 = -0.3014$, $p = 0.05$			
ξ_{Cl}	-0.01950	0.00079	0.00117	
ξ_{HL}	-0.01874	0.00075	0.00110	0.94178
ξ_{WX}	-0.03648	0.00157	0.00290	2.47912
ξ_U	-0.04100	0.00244	0.00412	3.52278
	$\sigma^2 = 0.05$, $k_2 = 0.531$, $k_3 = -0.5293$, $p = 0.05$			
ξ_{Cl}	-0.04357	0.00395	0.00584	
ξ_{HL}	-0.04188	0.00375	0.00550	0.94178
ξ_{WX}	-0.08153	0.00784	0.01449	2.47908
ξ_U	-0.09165	0.01219	0.02059	3.52300
	$\sigma^2 = 0.1$, $k_2 = 0.751$, $k_3 = -0.7485$, $p = 0.05$			
ξ_{Cl}	-0.06165	0.00789	0.01169	
ξ_{HL}	-0.05926	0.00750	0.01101	0.94178
ξ_{WX}	-0.11534	0.01568	0.02899	2.47910
ξ_U	-0.12964	0.02438	0.04119	3.52281
	$\sigma^2 = 0.2$, $k_2 = 1.0621$, $k_3 = -1.0586$, $p = 0.05$			
ξ_{Cl}	-0.08718	0.01578	0.02338	
ξ_{HL}	-0.08380	0.01500	0.02202	0.94178
ξ_{WX}	-0.16311	0.03136	0.05797	2.47912
ξ_U	-0.18334	0.04876	0.08238	3.52289
	$\sigma^2 = 0.5$, $k_2 = 1.6793$, $k_3 = -1.6738$, $p = 0.05$			
ξ_{Cl}	-0.13783	0.03946	0.05846	
ξ_{HL}	-0.13248	0.03750	0.05505	0.94178
ξ_{WX}	-0.25789	0.07841	0.14492	2.47912
ξ_U	-0.28987	0.12191	0.20593	3.52291

Table 5.3: Table of Bias, Variance and D-optimality criteria for Model 1 extrapolation with $x_0 = 0.75$, and $n = 30$.

	RB (%)	PI CP	PI Avg. Length
	$\underline{\sigma^2 = 0.01}$		
ξ_{Cl}	-1.0514	100	2.3128
ξ_{HL}	-2.0442	100	3.0801
ξ_{WX}	-1.0926	100	2.3249
ξ_U	-2.2985	100	3.6743
	$\underline{\sigma^2 = 0.05}$		
ξ_{Cl}	-2.2960	100	2.1432
ξ_{HL}	-4.4605	100	2.9091
ξ_{WX}	-2.3887	100	2.1636
ξ_U	-5.0181	100	3.5079
	$\underline{\sigma^2 = 0.1}$		
ξ_{Cl}	-3.2027	100	2.0421
ξ_{HL}	-6.2124	100	2.8036
ξ_{WX}	-3.3347	100	2.0689
ξ_U	-6.9657	100	3.4048
	$\underline{\sigma^2 = 0.2}$		
ξ_{Cl}	-4.4090	100	1.9994
ξ_{HL}	-8.6117	100	2.7294
ξ_{WX}	-4.5889	100	2.0317
ξ_U	-9.6643	100	3.3263
	$\underline{\sigma^2 = 0.5}$		
ξ_{Cl}	-6.6700	100	2.6004
ξ_{HL}	-12.9484	99.988	3.2002
ξ_{WX}	-6.9327	100	2.6308
ξ_U	-14.5431	99.926	3.7529

Table 5.4: Table comparing relative bias, prediction interval coverage percentage and prediction interval average length between designs for Model 1 extrapolation with $x_0 = 0.75$, and $n = 30$.

	$Bias(\hat{y}(x_0))$	$Var(\hat{y}(x_0))$	$MSE(\hat{y}(x_0))$	Eff(ξ_{Cl},ξ)
	$\sigma^2 = 0.01$, $k_2 = 0.2437$, $k_3 = -0.0973$, $p = 0.05$			
ξ_{Cl}	0.03006	0.00876	0.00967	
ξ_{HL}	0.03000	0.00833	0.00923	0.95508
ξ_{WX}	0.02921	0.01949	0.02034	2.10376
ξ_{U}	0.02867	0.02372	0.02454	2.53865
	$\sigma^2 = 0.05$, $k_2 = 0.048$, $k_3 = -0.025$, $p = 0.05$			
ξ_{Cl}	0.08717	0.04382	0.05142	
ξ_{HL}	0.08700	0.04167	0.04924	0.95753
ξ_{WX}	0.08598	0.09743	0.10482	2.03851
ξ_{U}	0.08489	0.11860	0.12581	2.44674
	$\sigma^2 = 0.1$, $k_2 = 0.079$, $k_3 = -0.038$, $p = 0.05$			
ξ_{Cl}	0.09618	0.08764	0.09689	
ξ_{HL}	0.09600	0.08333	0.09255	0.95517
ξ_{WX}	0.09352	0.19485	0.20360	2.10127
ξ_{U}	0.09183	0.23720	0.24564	2.53514
	$\sigma^2 = 0.2$, $k_2 = 0.111$, $k_3 = -0.051$, $p = 0.05$			
ξ_{Cl}	0.09918	0.17528	0.18512	
ξ_{HL}	0.09900	0.16667	0.17647	0.95326
ξ_{WX}	0.09491	0.38970	0.39871	2.15378
ξ_{U}	0.09260	0.47441	0.48298	2.60901
	$\sigma^2 = 0.5$, $k_2 = 0.168$, $k_3 = -0.084$, $p = 0.05$			
ξ_{Cl}	0.25248	0.43821	0.50196	
ξ_{HL}	0.25200	0.41667	0.48017	0.95660
ξ_{WX}	0.24754	0.97426	1.03553	2.06300
ξ_{U}	0.24384	1.18602	1.24548	2.48125

Table 5.5: Table of Bias, Variance and D-optimality criteria for Model 1 extrapolation with $x_0 = 2.5$, and $n = 30$.

	RB (%)	PI CP	PI Avg. Length
	$\underline{\sigma^2 = 0.01}$		
ξ_{Cl}	0.8654	100	14.0225
ξ_{HL}	0.8452	100	18.0903
ξ_{WX}	0.8677	100	14.1888
ξ_U	0.8269	100	19.5686
	$\underline{\sigma^2 = 0.05}$		
ξ_{Cl}	2.5523	100	13.9791
ξ_{HL}	2.5055	100	18.0520
ξ_{WX}	2.5565	100	14.1472
ξ_U	2.5054	100	19.5349
	$\underline{\sigma^2 = 0.1}$		
ξ_{Cl}	2.8298	100	13.9427
ξ_{HL}	2.7555	100	18.0303
ξ_{WX}	2.8138	100	14.1087
ξ_U	2.7120	100	19.5043
	$\underline{\sigma^2 = 0.2}$		
ξ_{Cl}	2.9098	100	13.9291
ξ_{HL}	2.7910	100	18.0324
ξ_{WX}	2.9114	100	14.0990
ξ_U	2.7231	100	19.5059
	$\underline{\sigma^2 = 0.5}$		
ξ_{Cl}	7.6999	100	14.0469
ξ_{HL}	7.5816	99.914	18.2464
ξ_{WX}	7.8302	100	14.2424
ξ_U	7.4229	99.858	19.7293

Table 5.6: Table comparing relative bias, prediction interval coverage percentage and prediction interval average length between designs for Model 1 extrapolation with $x_0 = 2.5$, and $n = 30$.

5.4 Model 2: $\boldsymbol{z}^T(x) = (1, x, x^2)^T$

Determination of k's and p Consider the case of fitting Model 2 to the data (i.e. $Y^* = \beta_0^* + \beta_1^* x + \beta_2^* x^2 + \epsilon$), however the true model is actually $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + k_3\beta_3 x^3 + k_4\beta_4 x^4 + \epsilon$. We will let all true β values be set equal to 1, and we will solve for β_0^*, β_1^*, β_2^*, k_3 and k_4 using the following five equations:

$$\frac{\partial}{\partial \beta_i^*} \int_{-0.5}^{0.5} [f(x) - f^*(x)]^2 dx = 0, \quad i = 0, 1, 2,$$

$$\int_{-0.5}^{0.5} [f(x) - f^*(x)]^2 dx = \eta^2, \quad \text{and}$$

$$\int_{0.5}^{x_0} [f(x) - f^*(x)]^2 dx = \eta^2.$$

The resulting values of k_3 and k_4 (and β^*'s for interest) are shown in Table 5.7 for $x_0 = 0.75$ and in Table 5.8 for $x_0 = 2.5$. The optimal p values with minimum value taken to be $p = 0.05$ are calculated in the same way as in previous sections, except using the values k_3 and k_4 in the true function found in the previous step.

σ^2	$\frac{\sigma^2}{n}$	β_0^*	β_1^*	β_2^*	k_3	k_4	p
0.01	0.0004	0.9901	0.8674	1.3972	-0.8842	1.8538	0.05
0.05	0.0018	0.9778	0.7034	1.8883	-1.9771	4.1452	0.05
0.10	0.0036	0.9686	0.5806	2.2562	-2.7961	5.8622	0.05
0.20	0.0071	0.9556	0.4069	2.7765	-3.9543	8.2904	0.05
0.50	0.0179	0.9298	0.0622	3.8089	-6.2522	13.1083	0.05

Table 5.7: Values of k_3, k_4 and p when fitting Model 2 for extrapolation at $x_0 = 0.75$, with $n = 28$.

σ^2	$\frac{\sigma^2}{n}$	β_0^*	β_1^*	β_2^*	k_3	k_4	p
0.01	0.0002	1.0000	1.0005	0.9996	0.0030	-0.0020	0.05
0.05	0.0010	1.0000	1.0033	0.9981	0.0220	-0.0090	0.05
0.10	0.0020	1.0001	1.0053	0.9964	0.0350	-0.0170	0.05
0.20	0.0041	1.0001	1.0080	0.9951	0.0530	-0.0230	0.05
0.50	0.0102	1.0002	1.0122	0.9916	0.0810	-0.0390	0.05

Table 5.8: Values of k_3, k_4 and p when fitting Model 2 for extrapolation at $x_0 = 2.5$, with $n = 49$.

Comparison of Designs To compare designs we can compute the asymptotic extrapolation bias, extrapolation variance and extrapolation MSE for each

design using the formulas described in Section 5.1. Table 5.9 shows a comparison of the four designs when $x_0 = 0.75$ and Table 5.11 shows the comparison when $x_0 = 2.5$.

Table 5.4 shows relative bias, prediction interval coverage percentages, and prediction interval average length for Model 2, when $x_0 = 0.75$. Table 5.6 shows the values of interest for each of the four designs for Model 2 when $x_0 = 2.5$.

	$Bias(\hat{y}(x_0))$	$Var(\hat{y}(x_0))$	$MSE(\hat{y}(x_0))$	Eff(ξ_{Cl},ξ)
	$\sigma^2 = 0.01$, $k_3 = -0.8842$, $k_4 = 1.8538$, $p = 0.05$			
ξ_{Cl}	-0.12150	0.00476	0.01953	
ξ_{HL}	-0.11863	0.00437	0.01845	0.94483
ξ_{WX}	-0.11103	0.00843	0.02076	1.06312
ξ_U	-0.09255	0.01512	0.02368	1.21291
	$\sigma^2 = 0.05$, $k_3 = -1.9771$, $k_4 = 4.1452$, $p = 0.05$			
ξ_{Cl}	-0.27169	0.02381	0.09763	
ξ_{HL}	-0.26527	0.02187	0.09224	0.94483
ξ_{WX}	-0.24827	0.04215	0.10379	1.06312
ξ_U	-0.20696	0.07558	0.11841	1.21291
	$\sigma^2 = 0.1$, $k_3 = -2.7961$, $k_4 = 5.8622$, $p = 0.05$			
ξ_{Cl}	-0.38421	0.04763	0.19524	
ξ_{HL}	-0.37513	0.04375	0.18447	0.94483
ξ_{WX}	-0.35109	0.08430	0.20757	1.06312
ξ_U	-0.29266	0.15116	0.23682	1.21292
	$\sigma^2 = 0.2$, $k_3 = -3.9543$, $k_4 = 8.2904$, $p = 0.05$			
ξ_{Cl}	-0.54335	0.09525	0.39048	
ξ_{HL}	-0.53051	0.08750	0.36894	0.94483
ξ_{WX}	-0.49651	0.16861	0.41513	1.06312
ξ_U	-0.41388	0.30233	0.47363	1.21293
	$\sigma^2 = 0.5$, $k_3 = -6.2522$, $k_4 = 13.1083$, $p = 0.05$			
ξ_{Cl}	-0.85914	0.23813	0.97626	
ξ_{HL}	-0.83883	0.21875	0.92239	0.94483
ξ_{WX}	-0.78508	0.42152	1.03788	1.06312
ξ_U	-0.65444	0.75582	1.18411	1.21291

Table 5.9: Table of Bias, Variance and D-optimality criteria for Model 2 extrapolation with $x_0 = 0.75$, and $n = 28$.

	RB (%)	PI CP	PI Avg. Length
	$\underline{\sigma^2 = 0.01}$		
ξ_{Cl}	-4.6965	100	5.6163
ξ_{HL}	-4.3964	100	6.9312
ξ_{WX}	-4.8076	100	5.7197
ξ_U	-3.6642	100	9.3063
	$\underline{\sigma^2 = 0.05}$		
ξ_{Cl}	-9.5079	100	5.9972
ξ_{HL}	-8.9026	100	7.4581
ξ_{WX}	-9.7391	100	6.1035
ξ_U	-7.4206	100	10.0737
	$\underline{\sigma^2 = 0.1}$		
ξ_{Cl}	-12.5635	100	6.3191
ξ_{HL}	-11.7400	100	7.8908
ξ_{WX}	-12.8330	100	6.4315
ξ_U	-9.8175	100	10.6764
	$\underline{\sigma^2 = 0.2}$		
ξ_{Cl}	-16.2541	100	6.8371
ξ_{HL}	-15.1682	100	8.5664
ξ_{WX}	-16.6343	100	6.9459
ξ_U	-12.7427	100	11.5798
	$\underline{\sigma^2 = 0.5}$		
ξ_{Cl}	-21.9277	99.854	8.1970
ξ_{HL}	-20.4841	99.554	10.2305
ξ_{WX}	-22.4743	99.758	8.3004
ξ_U	-17.1186	99.462	13.7501

Table 5.10: Table comparing relative bias, prediction interval coverage percentage and prediction interval average length between designs for Model 2 extrapolation with $x_0 = 0.75$, and $n = 28$.

	$Bias(\hat{y}(x_0))$	$Var(\hat{y}(x_0))$	$MSE(\hat{y}(x_0))$	Eff(ξ_{Cl},ξ)
	$\sigma^2 = 0.01$, $k_3 = 0.003$, $k_4 = -0.002$, $p = 0.05$			
ξ_{Cl}	0.03004	0.52429	0.52519	
ξ_{HL}	0.03000	0.49000	0.49090	0.93470
ξ_{WX}	0.02979	0.80359	0.80447	1.53177
ξ_U	0.02965	1.30248	1.30336	2.48167
	$\sigma^2 = 0.05$, $k_3 = 0.022$, $k_4 = -0.009$, $p = 0.05$			
ξ_{Cl}	0.00750	2.62145	2.62151	
ξ_{HL}	0.00750	2.45000	2.45006	0.93460
ξ_{WX}	0.00505	4.01793	4.01796	1.53269
ξ_U	0.00392	6.51238	6.51240	2.48422
	$\sigma^2 = 0.1$, $k_3 = 0.035$, $k_4 = -0.017$, $p = 0.05$			
ξ_{Cl}	0.11264	5.24290	5.25559	
ξ_{HL}	0.11250	4.90000	4.91266	0.93475
ξ_{WX}	0.10903	8.03587	8.04776	1.53128
ξ_U	0.10727	13.02477	13.03627	2.48046
	$\sigma^2 = 0.2$, $k_3 = 0.053$, $k_4 = -0.023$, $p = 0.05$			
ξ_{Cl}	0.06757	10.48581	10.49037	
ξ_{HL}	0.06750	9.80000	9.80456	0.93462
ξ_{WX}	0.06180	16.07174	16.07556	1.53241
ξ_U	0.05909	26.04954	26.05303	2.48352
	$\sigma^2 = 0.5$, $k_3 = 0.081$, $k_4 = -0.039$, $p = 0.05$			
ξ_{Cl}	0.24780	26.21451	26.27592	
ξ_{HL}	0.24750	24.50000	24.56126	0.93474
ξ_{WX}	0.23941	40.17934	40.23666	1.53131
ξ_U	0.23533	65.12384	65.17922	2.48057

Table 5.11: Table of Bias, Variance and D-optimality criteria for Model 2 extrapolation with $x_0 = 2.5$, and $n = 49$.

	RB (%)	PI CP	PI Avg. Length
	$\underline{\sigma^2 = 0.01}$		
ξ_{Cl}	0.3065	100	252.0512
ξ_{HL}	0.3084	100	322.2777
ξ_{WX}	0.3042	100	260.6871
ξ_U	0.2956	100	412.5136
	$\underline{\sigma^2 = 0.05}$		
ξ_{Cl}	0.0963	100	252.1437
ξ_{HL}	0.0540	100	322.2382
ξ_{WX}	0.0836	100	260.7532
ξ_U	0.0325	100	412.4250
	$\underline{\sigma^2 = 0.1}$		
ξ_{Cl}	1.1987	100	252.0665
ξ_{HL}	1.1570	100	322.1388
ξ_{WX}	1.1968	100	260.7029
ξ_U	1.1311	100	412.2851
	$\underline{\sigma^2 = 0.2}$		
ξ_{Cl}	0.7833	100	252.3321
ξ_{HL}	0.6667	100	322.2228
ξ_{WX}	0.7872	100	261.0005
ξ_U	0.5752	99.998	412.0900
	$\underline{\sigma^2 = 0.5}$		
ξ_{Cl}	2.5375	99.944	252.7642
ξ_{HL}	2.3304	99.996	325.5544
ξ_{WX}	2.7048	99.964	262.0025
ξ_U	2.9959	100	430.5651

Table 5.12: Table comparing relative bias, prediction interval coverage percentage and prediction interval average length between designs for Model 2 extrapolation with $x_0 = 2.5$, and $n = 49$.

5.5 Discussion

Linear Model Unfortunately, in extrapolation scenarios for Model 1 for $x_0 = 0.75$ and 2.5 we see that the clustered design has higher bias and variance than those for the HL design. HL design is consistently the best design to use. Both of these designs outperform the WX and uniform design as well. We see the same results when using the simulated bias, variance and MSE instead of asymptotic.

The clustered design outperforms the others in relative bias, however this is only when $x_0 = 0.75$ (closer to the sample space). As x_0 gets further away from the sample space, the clustered design no longer is the best and the Hoel-Levine design remains optimal. Overall, for extrapolation when fitting the simple-linear model, the Hoel-Levine design stays the most optimal.

Quadratic Model For Model 2, we see many of the same results. Bias is not improved by using the clustered design, and the HL design remains Q-optimal and has lowest bias and variance. Again, we see the same results when using the simulated bias, variance and MSE instead of asymptotic. HL design and uniform have better relative bias and coverage percentage for both extrapolation points.

Once again, the Hoel-Levine design is still the most optimal for fitting the quadratic model in an extrapolation scenario.

5.6 Set-up of clustered design for extrapolation

The clustered design can be set up in many different ways for extrapolation. Every method will have m areas located at each design support point of the corresponding HL design with clusters of distinct design points distributed across them.

- Method 1 be where the areas at each HL design point have size p/m, and the number of distinct design points clustered over each area is nw_i, where w_i's are the weights calculated for the HL design.
- Method 2 use areas of size pw_i and the number of distinct design points clustered over each area is n/m.
- Method 3 will have areas of size pw_i and the number of points on each cluster will be nw_i.

Tables 5.13 and 5.14 show a comparison of the SMSE between the different methods of setting up the clustered design for Model 1 at the two extrapolation points. $N = 5000$ was used for this method-comparison.

σ^2	Method 1	Method 2	Method 3
0.01	0.000013	0.000028	0.000024
0.05	0.000221	0.000355	0.000285
0.10	0.000824	0.001296	0.000974
0.20	0.003292	0.004669	0.003535
0.50	0.019721	0.029840	0.020400

Table 5.13: Comparison of SMSE's for different methods of constructing clustered design for extrapolation for Model 1 when $x_0 = 0.75$

σ^2	Method 1	Method 2	Method 3
0.01	0.000092	0.000120	0.000121
0.05	0.002265	0.002402	0.002392
0.10	0.008626	0.009595	0.009317
0.20	0.034303	0.034954	0.036127
0.50	0.222105	0.230657	0.222230

Table 5.14: Comparison of SMSE's for different methods of constructing clustered design for extrapolation for Model 2, $x_0 = 2.5$

We can see that overall Method 1 of constructing the clustered design worked slightly better than the other methods. Therefore, this is the way that clustered design was constructed in this study.

Chapter 6

Conclusions

Recommendations

For a static design process (only one design) the clustered design may be used as a replacement for the uniform design if some previous knowledge of the true model is known. Often we saw the clustered design having a much higher efficiency than uniform. But, if the experimenter has no prior information about the true model, then uniform design should still be used as normal. In high dimensional data however, the clustered design is not recommended to be implemented.

For a multi-stage design process, we may use the clustered design in the first stage of the design, again if some prior knowledge of the true model is known. Otherwise, uniform should continue to be used in the first stage. In the second stage, if we conduct a hypothesis test for a model and the test is accepted, the optimal design to use is the classical D-optimal design. If the hypothesis test fails, then the clustered design may be used as a replacement for Huber's robust design or Weins-Xu's design as a new robust design, since we saw that clustered design also outperformed Huber's and WX designs in terms of efficiency.

The clustered design would require more work to be implemented effectively in extrapolation scenarios as well. While it did have some merit in measurements such as relative bias and prediction interval average length, the efficiency was still not as good as Hoel-Levine's design. In extrapolation scenarios, we recommend continuing to use the HL design when the hypothesis is accepted for the fitted model.

Future Studies

Some areas of future study for the clustered design:

1. Using the clustered design in bivariate or higher dimensional data.
2. Using the clustered design in extrapolation scenarios.

Bibliography

[1] Fedorov, V. V. (1972). Thoery of Optimal Experiments. Academic Press Inc. New York.

[2] Heo, G. (1998). Optimal Designs for Approximately Polynomial Regression Models. *University of Alberta*.

[3] Heo, G. (2001). Restricted minimax robust designs for misspecified regression models. In: *The Canadian Journal of Statistics*. Vol. 29, No. 1, 117-128.

[4] Hoel, P.G. and Levine, A. (1964). Optimal spacing and weighting in polynomial prediction. *Ann. Math. Statist.* Vol. 35, 1553-1560.

[5] Huber, P.J. (1975). Robustness and Designs. In: *A Survey of Statistical Design and Linear Models.* ed. Strivastava, J.N., Amsterdam: North Holland, 287-303.

[6] Huber, P.J. (1981). Robust statistics. John Wiley & Sons. New York.

[7] Pukelsheim, F. (1992). Efficient rounding of approximate designs. In: *Biometrica*. Vol. 79, No. 4, 763-770.

[8] Wiens, D.P. (2019). Cluster, Don't Replicate.

https://sites.ualberta.ca/ dwiens/home page/techReports/clustering.pdf.

[9] Wiens, D.P. (1992). Minimax designs for approximately linear regression. *Journal of Statistical Planning and Inference*. Vol. 31, 353-371.

[10] Wiens, D.P. and Xu, X. (2008). Robust designs for one-point extrapolation. *Journal of Statistical Planning and Inference*. Vol. 138, 1339-1357.

[11] Xu, X. and Yuen, W.K. (2011). Applications and Implementations of Continuous Robust Designs. *Communications in Statistics - Theory and Methods*. Vol. 40, No. 6, 969-988.

Appendices

Appendix I: Notations

- ALT: Accelerated Life Testing
- CDD: Classical D-optimal Design
- $d(x)$ = density function used to calculate design points for HRD.
- Eff(ξ_1, ξ_2) = Relative efficiency of design 1 compared to design 2
- $f(x)$ = true mean response function
- $f^*(x)$ = fitted mean response function
- $g(x)$ = contamination function
- HRD: Huber's implemented Robust Design
- k = the estimated maximum true value of β_2 such that $H_0 : \beta_2 = 0$ is accepted
- m = number of support points for a design
- $MSE(\hat{\boldsymbol{\beta}})$ = mean squared error of OLS estimator $\hat{\boldsymbol{\beta}}$
- n = sample size for simulations
- N = number of trials/iterations for simulations
- OLS: Ordinary Least-Squares
- p = percentage of design space covered by Clustered Design
- PI: prediction interval
- S = Design Space
- X = design matrix
- x_0 = an extrapolation point

- $\boldsymbol{y}$ = response vector
- Y = true model
- Y^* = fitted model
- $\boldsymbol{z}(\boldsymbol{x})$ = the fitted model to the data represented as a vector
- $\hat{\boldsymbol{\beta}}$ = the estimated parameters in the regression model
- $\nu = \frac{\sigma^2}{n\eta^2} \in (0, \infty)$ = the relative belief of model accuracy
- η^2 = bound for contamination in model, found in this report using $\nu = 1$ from the equation for ν
- $\xi_{CDD}^{(2)}$ = classical D-optimal design assuming true model is linear
- $\xi_{CDD}^{(3)}$ = classical D-optimal design assuming true model is quadratic
- $\xi_{CDD}^{(4)}$ = classical D-optimal design assuming true model is cubic
- $\xi_H^{(m)}$ = implemented Huber's design with m distinct design points.
- ξ_U = uniform design
- ξ_{Cl} = clustered design

Appendix II: R Code

The following section lists the R codes used in the simulation. Some codes have been left out due to high similarity in order to reduce the report length. For instance, for regression there was a different R file for each model, but only Model1.R will be listed. All codes including the ones exempted here can be found on Github at this link: https://github.com/.

Summary (README)

- Model1.R is the source script which runs entire regression simulation for Model 1. Model2.R and Model3.R (found on github) are similar but for Model 2 and 3 respectively. Model4.R, Model5.R and Model6.R (also found on github) run the simulation where the contamination function is more complex for Models 1, 2 and 3 respectively.
- kaccept.R finds the k value for each Model in Chapter 3.
- kL2.R finds the k values for each Model for Chapter 4: Complex Contamination functions.
- Optp.R finds the D-optimal p value for the clustered design.
- Computexj.R finds the Huber's implemented design points.
- resultstable.R compiles the results for each design into a table.
- MSEdesign.R runs the actual simulation for a given design for the parameter estimation simulation.
- RunEP.R is the source script which is executed to run the simulation for both models and both x_0's.
- Extrapolation.R runs the entire simulation for whichever model and x_0 is selected.
- EPfunctions.R contains all the functions used for the extrapolation simulation. The functions that it contains are highly similar in concept to the functions used for regression analysis. All functions for extrapolation were contained in one .R file instead.
- 2.5kvalsmodel1.R and 2.5kvalsmodel2.R are the functions used to determine the k values used when extrapolating at $x_0 = 2.5$ for both models, since solving the equations exactly resulted in imaginary numbers, so the next best option was found analytically with these two functions.
- CIrun.R runs the entire simulation for all the extra measurements such as relative bias, confidence intervals, sim SE, etc... for both models and both x_0's.

- CIcomparison.R runs the simulation for a given model and x_0 and compiles the results into a table.
- CIfunction.R contains all functions used in the extra measurement simulations, which are again highly similar to the functions used for regression and for extrapolation.

Regression Codes

Model1.R

```
#Model 1: Simple-Linear Model

cat('\f')
rm(list=ls())
#Retrieve functions
setwd('C:/Users/dburm/Desktop/School/MATH 4F90/')
source('MSEdesign.R')
source('Computexj.R')
source('kaccept.R')
source('Optp.R')
source('resultstable.R')
source('kL2.R')
library(xtable)


#####Determine k and p
n <- 30
N <- 50000
model <- 1
#Create vector of sigma's to be tested
sigma2vector <- c(0.01,0.05,0.1,0.2,0.5)

results <- matrix(0,nrow = length(sigma2vector),ncol = 6)
colnames(results) <- c("Sigma^2","12Sigma^2/n","k-C","p-C","k-U","p-U")
results[,1] <- sigma2vector
results[,2] <- 12*results[,1]/n
X <- cbind(rep(1,n),c(rep(-0.5,n/3),rep(0,n/3),rep(0.5,n/3)))
results[,3] <- kaccept(X,sigma2vector,n,N = 100,maxBplus = 2,step = 0.02,model = model)
results[,4] <- Optp(n,N=5000,sigma2vector,kvector = results[,3],model = model,ktype="kC")
X <- cbind(rep(1,n),seq(-0.5,0.5,by=1/(n-1)))
results[,5] <- kaccept(X,sigma2vector,n,N = 100,maxBplus = 3,step = 0.02,model = model)
results[,6] <- Optp(n,N=5000,sigma2vector,kvector = results[,5],model = model,ktype="kU")


###Design results tables
#k-C
```

```
#Classical D-Opt Design
X <- cbind(rep(1,n),c(rep(-0.5,n/2),rep(0.5,n/2)))
resultsCDD.1 <- resultstable(X,n,N,sigma2vector,kvector=results[,3],Cldesign = F,
                             pvector = F,model = model)

#Huber's Implemented Design
X <- cbind(rep(1,n),c(rep(-0.396,n/3),rep(0,n/3),rep(0.396,n/3)))
resultsH3.1 <- resultstable(X,n,N,sigma2vector,kvector=results[,3],Cldesign = F,
                             pvector = F,model = model)

#Uniform Design
X <- cbind(rep(1,n),seq(-0.5,0.5,by=1/(n-1)))
resultsU.1 <- resultstable(X,n,N,sigma2vector,kvector=results[,3],Cldesign = F,
                             pvector = F,model = model)

#Clustered Design for model 1
resultsCl.1 <- resultstable(X=F,n,N,sigma2vector,kvector=results[,3],Cldesign = T,
                             pvector = results[,4],model = model)

#Tables of interest
BiasVarTable.1 <- matrix(NA,nrow = length(sigma2vector)*5,ncol = 5)
colnames(BiasVarTable.1) <- c("BiasB0","BiasB1","VarB0","VarB1","det(MSE)")
for (i in 1:length(sigma2vector)){
  BiasVarTable.1[5*(i-1)+1, 1:4] <- c(results[i,c(1,3,4)],N)
  BiasVarTable.1[5*(i-1)+2, 1:5] <- c(resultsCDD.1[i,c(3,4,5,6,8)])
  BiasVarTable.1[5*(i-1)+3, 1:5] <- c(resultsH3.1[i,c(3,4,5,6,8)])
  BiasVarTable.1[5*(i-1)+4, 1:5] <- c(resultsCl.1[i,c(3,4,5,6,8)])
  BiasVarTable.1[5*i, 1:5] <- c(resultsU.1[i,c(3,4,5,6,8)])
}

EffTable.1 <- matrix(NA,nrow = length(sigma2vector),ncol = 6)
colnames(EffTable.1) <- c("sigma^2","k-C","p-C","CDD/CL","H3/CL","U/CL")
EffTable.1[,1] <- sigma2vector
EffTable.1[,2] <- results[,3]
EffTable.1[,3] <- results[,4]
EffTable.1[,4] <- resultsCDD.1[,8]/resultsCl.1[,8]
EffTable.1[,5] <- resultsH3.1[,8]/resultsCl.1[,8]
EffTable.1[,6] <- resultsU.1[,8]/resultsCl.1[,8]



#kU
#Classical D-Opt Design
X <- cbind(rep(1,n),c(rep(-0.5,n/2),rep(0.5,n/2)))
resultsCDD.2 <- resultstable(X,n,N,sigma2vector,kvector=results[,5],Cldesign = F,
                             pvector = F,model = model)

#Huber's Implemented Design
X <- cbind(rep(1,n),Computexj(m = 30,model = 1))
resultsH30.2 <- resultstable(X,n,N,sigma2vector,kvector = results[,5],Cldesign = F,
```

```
                            pvector = F,model = model)

#Uniform Design
X <- cbind(rep(1,n),seq(-0.5,0.5,by=1/(n-1)))
resultsU.2 <- resultstable(X,n,N,sigma2vector,kvector=results[,5],Cldesign = F,
                            pvector = F,model = model)

#Clustered Design for model 1
resultsCl.2 <- resultstable(X=F,n,N,sigma2vector,kvector=results[,5],Cldesign = T,
                            pvector = results[,6],model = model)

#Tables of interest
BiasVarTable.2 <- matrix(NA,nrow = length(sigma2vector)*5,ncol = 5)
colnames(BiasVarTable.2) <- c("BiasB0","BiasB1","VarB0","VarB1","det(MSE)")
for (i in 1:length(sigma2vector)){
  BiasVarTable.2[5*(i-1)+1, 1:4] <- c(results[i,c(1,5,6)],N)
  BiasVarTable.2[5*(i-1)+2, 1:5] <- c(resultsCDD.2[i,c(3,4,5,6,8)])
  BiasVarTable.2[5*(i-1)+3, 1:5] <- c(resultsH30.2[i,c(3,4,5,6,8)])
  BiasVarTable.2[5*(i-1)+4, 1:5] <- c(resultsCl.2[i,c(3,4,5,6,8)])
  BiasVarTable.2[5*i, 1:5] <- c(resultsU.2[i,c(3,4,5,6,8)])
}

EffTable.2 <- matrix(NA,nrow = length(sigma2vector),ncol = 6)
colnames(EffTable.2) <- c("sigma^2","k-U","p-U","CDD/CL","H30/CL","U/CL")
EffTable.2[,1] <- sigma2vector
EffTable.2[,2] <- results[,5]
EffTable.2[,3] <- results[,6]
EffTable.2[,4] <- resultsCDD.2[,8]/resultsCl.2[,8]
EffTable.2[,5] <- resultsH30.2[,8]/resultsCl.2[,8]
EffTable.2[,6] <- resultsU.2[,8]/resultsCl.2[,8]


#Save results as RDS
model1results <- list(BiasVarTablekC = BiasVarTable.1,BiasVarTablekU = BiasVarTable.2,
                      EffTablekC = EffTable.1,EffTablekU = EffTable.2,
                      results = results,
                      resultsCDDkC = resultsCDD.1,resultsCDDkU = resultsCDD.2,
                      resultsClkC = resultsCl.1,resultsClkU = resultsCl.2,
                      resultsH3kC = resultsH3.1,resultsH30kU = resultsH30.2,
                      resultsUkC = resultsU.1,resultsUkU = resultsU.2)
saveRDS(model1results,file = paste("model",model,"results.rds",sep=""))
```

kaccept.R

```
    #Finds the max value of k for which Ho: B2 = 0 is accepted for all sigmas inputted

#Inputs
#model = 1: Linear
#model = 2: quadratic
```

```
#model = 3: MLR

#model = 1 or 2:
#X: nx2 matrix of vectors (1,x)
#model = 3:
#X: nx3 matrix of vectors (1,x1,x2)

#n: sample size
#N: Number of hypothesis tests
#sigma2vector: vector of sigma^2's (variance) to find k for
#model 1: maxBplus is max true value of B2 to test
#model 2: maxBplus is max true value of B3 to test
#model 3: maxBplus is max true value of B12 to test
#step: amount to decrease from maxBplus by until k is found


#Output
#kresults: value of k for each sigma in sigmavector

kaccept <- function(X,sigma2vector,n,N,maxBplus,step,model){
  B0 <- 1
  B1 <- 1
  B2 <- 1

  if (model == 1){
    X <- cbind(X,X[,2]^2)
  }else if (model == 2){
    X <- cbind(X,X[,2]^2,X[,2]^3)
  }else if (model == 3){
    X <- cbind(X,X[,2]*X[,3])
  }

  #Create results vector
  kresults <- vector(mode="double",length = length(sigma2vector))

  for (joe in 1:100){
    print(joe)
    #Count variable for inputting results
    sigmacount <- 0
    #Start looping through sigma^2 values
    for (sigma2 in sigma2vector){
      sigmacount <- sigmacount + 1

      #Loop through k values starting large and working way down until H0 is accepted
      for (k in seq(maxBplus,0,by=-step)){
        #k is the true value of B2
        Bplus <- k

        #Responses for each N trial
        accept <- vector(mode="double",length=N)
```

```
        #Loop through N hypothesis tests of this comb. of X,k and sigma
        for (j in 1:N){
          #Create y (response) vector
          y <- vector(mode="double",length = n)

          #Generate simulated data and values needed for hypothesis test based on model chosen
          if (model == 1){
            for (i in 1:n){
              y[i] <- B0 + B1*X[i,2] + Bplus*X[i,3] + rnorm(1, mean = 0, sd = sqrt(sigma2))
            }
            Bhat <- solve(t(X)%*%X)%*%t(X)%*%y
            CovB <- sigma2*solve(t(X)%*%X)
            VarBplushat <- CovB[3,3]
            Bplushat <- Bhat[3]
          }else if (model == 2|model == 3){
            for (i in 1:n){
              y[i] <- B0 + B1*X[i,2] + B2*X[i,3] + Bplus*X[i,4] +
                        rnorm(1, mean = 0, sd = sqrt(sigma2))
            }
            Bhat <- solve(t(X)%*%X)%*%t(X)%*%y
            CovB <- sigma2*solve(t(X)%*%X)
            VarBplushat <- CovB[4,4]
            Bplushat <- Bhat[4]
          }

          #If Ho is rejected, indicate with a 1
          if (abs(Bplushat) <= 1.96*sqrt(VarBplushat)){
            accept[j] <- 1
          }
        }

        #Check if Ho was rejected less than 95% of times
        #If so, we have found the max k value and exit loop
        if (sum(accept) > 0.95*N){
          kresults[sigmacount] <- kresults[sigmacount] + k
          break
        }
      }
    }
  }
  kresults <- kresults/100

  return(kresults)
}
```

Optp.R

```
    #Find optimal p values of clustered design for models:

#Inputs:
```

```
#model = 1: Linear
#model = 2: quadratic
#model = 3: MLR        ***n MUST be 36 for model = 3

#n: sample size
#N: Number of iterations (for estimating Bhat) to be submitted into MSEdesign function
#sigma2vector: vector of sigma^2's (variance) to find p for
#kvector: vector of k's found using kaccept.R function
#ktype: string for the saved-graph file name


Optp <- function(n,N,sigma2vector,kvector,model,ktype){

  prange <- seq(0.05,0.95,by=0.01)
  opt <- vector(mode="double",length = length(sigma2vector))
  #Matrix for storing estimated MSE for each p for each sigma^2
  pMSE <- matrix(0,nrow = length(prange), ncol = length(sigma2vector))
  colnames(pMSE) <- c("0.01","0.05","0.1","0.2","0.5")
  for (joe in 1:10){
    print(joe)
    for (j in 1:length(prange)){
      p <- prange[j]
      print(p)

      #Set up clustered design matrix
      if (model == 1|model == 4){
        X <- cbind(rep(1,n),c(seq(-0.5,-0.5 + p/2,by=(p/2)/(n/2-1)),
                              seq(0.5 - p/2,0.5,by=(p/2)/(n/2-1))))
      }else if (model == 2|model == 5){
        X <- cbind(rep(1,n),c(seq(-0.5,-0.5+p/3,by = p/3/(n/3-1)),
                              seq(0-p/6,0+p/6,by = p/3/(n/3-1)),
                              seq(0.5-p/3,0.5,by = p/3/(n/3-1))))
      }else if (model == 3|model == 6){
        X <- cbind(rep(1,n),
                   c(rep(0.5,6),rep(0.5-0.25*sqrt(p),6),rep(0.5-0.5*sqrt(p),6),
                     rep(-0.5+0.5*sqrt(p),6),rep(-0.5+0.25*sqrt(p),6),rep(-0.5,6)),
                   rep(c(0.5,0.5-0.25*sqrt(p),0.5-0.5*sqrt(p),-0.5+0.5*sqrt(p),
                         -0.5+0.25*sqrt(p),-0.5),6)
        )
      }
      #Estimate det(MSE) for each sigma^2 for this p
      for (i in 1:length(sigma2vector)){
        pMSE[j,i] <- pMSE[j,i] + MSEdesign(X,n,N,sigma2vector[i],kvector[i],model = model)[[1]]
      }
    }
  }
  pMSE <- pMSE/10

  #Plots
  for(i in 1:length(sigma2vector)){
```

```
    plot(prange,pMSE[,i],xlab = "p",ylab = "det(MSE(betahat))", pch = 19,col = "gray45",
         main = paste("Model ",model,".  sigma^2 =",sigma2vector[i],", k =",kvector[i]))

    fhat <- ksmooth(prange,pMSE[,i],kernel = "normal", bandwidth = 0.15,
                    range.x = range(prange),n.points = 100, x.points = prange)
    lines(fhat)
    dev.copy(png,paste("model",model,"s",sigma2vector[i],ktype,".png",sep=""))
    dev.off()

    opt[i] <- fhat$x[which.min(fhat$y)]
  }

  return(opt)
}
```

Computexj.R

```
    #Find xj for Huber's implemented design

#Inputs:
#model = 1: Linear
#model = 2: quadratic
#model = 3: MLR

#m: number of distinct design points

#Output:
#design points

Computexj <- function(m,model){

  if (model == 4){
    model <- 1
  }else if (model == 5){
    model <- 2
  }else if (model == 6){
    model <- 3
  }


  #####LINEAR AND QUADRATIC INTEGRATION
  if (model == 1|model == 2){
    if (model == 1){
      f <- function(x){
        5.12*x^2 + 0.573
      }
    }else if (model == 2){
      f <- function(x){
        35.0934*((x^2-0.1487^2)*(x^2-0.1489^2)+0.0192)
      }
```

```
    }

    h <- 0.0001
    xj <- vector(mode="double",length = m)

    for(j in 1:m){
      i <- -0.5
      x <- 0
      while(x < (j - 0.5)/m){
        x <- integrate(f,-0.5,i)$value
        i <- i + h
      }
      xj[j] <- i
    }
  #####MLR DOUBLE INTEGRATION
  }else if (model == 3){
    h <- 0.0001
    xj <- matrix(NA,nrow = m,ncol = 2)

    for(j in 1:(m/4)){
      i <- -0.5
      x <- 0
      while(x < (j - 0.5)/m){
        x <- integrate(function(y) {
          sapply(y,function(y){
            integrate(function(x){
              sapply(x,function(x){181.02*pmax((x^2-0.2333^2+0.044)+(y^2-0.2333^2+0.044),0)} )
            },-0.5,y)$value
          })
        },-0.5,i)$value
        i <- i + h
      }
      xj[m - j + 1,] <- c(i,i)
    }

    #Solve for remaining design points using symmetry
    if (m == 4){
      xj[1,] <- c(-i,-i)
      xj[2,] <- c(-i,i)
      xj[3,] <- c(i,-i)
    }else if (m > 4){
      xj[1:(m/4),] <- apply(-xj[(3*m/4+1):m,],2,rev)
      xj[(m/4+1):(2*m/4),] <- cbind(xj[1:(m/4),1],rev(xj[(3*m/4+1):m,2]))
      xj[(2*m/4+1):(3*m/4),] <- cbind(xj[(3*m/4+1):m,1],rev(xj[1:(m/4),2]))
    }

  }

  return(xj)
}
```

resultstable.R

```
    #Makes results table for designs

#Inputs:
#Cldesign: boolean stating whether its for the cluster design (T) or not (F)
#If Cldesign is false pvector can be anything
#If Cldesign is true then X can be anything
#Other variables are the same as other functions

resultstable <- function(X,n,N,sigma2vector,kvector,Cldesign,pvector,model){
  #####For linear model since table has less columns than model 2 or 3
  if (model == 1|model == 4){
    results <- matrix(NA,nrow = length(sigma2vector),ncol = 8)
    colnames(results) <- c("Sigma^2","k","BiasBo","BiasB1","VarB0","VarB1",
                           "det(Cov)","det(MSE)")
    results[,1] <- sigma2vector
    results[,2] <- kvector
    for (i in 1:length(sigma2vector)){
      if (Cldesign == T){
        p <- pvector[i]
        X <- cbind(rep(1,n),c(seq(-0.5,-0.5 + p/2,by=(p/2)/(n/2-1)),
                              seq(0.5 - p/2,0.5,by=(p/2)/(n/2-1))))
      }
      design <- MSEdesign(X,n,N,sigma2 = sigma2vector[i],k = kvector[i],model = model)
      results[i,3] <- design[[3]][[1]] #BiasB0
      results[i,4] <- design[[3]][[2]] #BiasB1
      results[i,5] <- design[[4]][[1]] #VarB0
      results[i,6] <- design[[4]][[2]] #VarB1
      results[i,7] <- design[[2]] #det(cov)
      results[i,8] <- design[[1]] #det(MSE)
    }
    if (Cldesign == T){
      results <- cbind(results,pvector)
      colnames(results)[9] <- "p"
    }
    ##### For quadratic and MLR models
  }else if (model == 2|model == 3|model == 5|model == 6){
    results <- matrix(0,nrow = length(sigma2vector),ncol = 10)
    colnames(results) <- c("Sigma^2","k","BiasB0","BiasB1","BiasB2","VarB0","VarB1","VarB2",
                           "det(Cov)","det(MSE)")
    results[,1] <- sigma2vector
    results[,2] <- kvector
    for (i in 1:length(sigma2vector)){
      if (Cldesign == T){
        p <- pvector[i]
        if (model == 2|model == 5){
          X <- cbind(rep(1,n),c(seq(-0.5,-0.5+p/3,by = p/3/(n/3-1)),
                                seq(0-p/6,0+p/6,by = p/3/(n/3-1)),
                                seq(0.5-p/3,0.5,by = p/3/(n/3-1))))
```

```
        }else if (model == 3|model == 6){
          X <- cbind(rep(1,n),
                     c(rep(0.5,6),rep(0.5-0.25*sqrt(p),6),rep(0.5-0.5*sqrt(p),6),
                       rep(-0.5+0.5*sqrt(p),6),rep(-0.5+0.25*sqrt(p),6),rep(-0.5,6)),
                     rep(c(0.5,0.5-0.25*sqrt(p),0.5-0.5*sqrt(p),-0.5+0.5*sqrt(p),
                           -0.5+0.25*sqrt(p),-0.5),6)  )
        }
      }
      design <- MSEdesign(X,n,N,sigma2 = sigma2vector[i],k = kvector[i],model = model)
      results[i,3] <- design[[3]][[1]] #BiasB0
      results[i,4] <- design[[3]][[2]] #BiasB1
      results[i,5] <- design[[3]][[3]] #BiasB2
      results[i,6] <- design[[4]][[1]] #VarB0
      results[i,7] <- design[[4]][[2]] #VarB1
      results[i,8] <- design[[4]][[3]] #VarB2
      results[i,9] <- design[[2]] #det(cov)
      results[i,10] <- design[[1]] #det(MSE)
    }
    if (Cldesign == T){
      results <- cbind(results,pvector)
      colnames(results)[11] <- "p"
    }
  }

  return(results)
}
```

MSEdesign.R

```
    #Inputs
#model = 1: Linear
#model = 2: quadratic
#model = 3: MLR

#model = 1 or 2:
#X: nx2 matrix of vectors (1,x)
#model = 3:
#X: nx3 matrix of vectors (1,x1,x2)

#model = 4,5,6 are the same as 1,2,3 except with 3 missing true parameters in regression model

#n: sample size
#N: Number of iterations (for estimating Bhat)
#sigma2: Variance of ei's
#model 1: k is true value of B2
#model 2: k is true value of B3
#model 3: k is true value of B12

#Outputs
#output: list containing biases, variances and determinants of interest
```

```
MSEdesign <- function(X,n,N,sigma2,k,model){
  B0 <- 1
  B1 <- 1

  #####LINEAR MODEL
  if (model == 1|model == 4){
    BetaN <- matrix(0,N,2)

    for (j in 1:N){
      #Simulate residuals and y values
      y <- vector(mode="double",length = n)
      if (model == 1){
        for (i in 1:n){
          y[i] <- B0 + B1*X[i,2] + k*X[i,2]^2 + rnorm(1, mean = 0, sd = sqrt(sigma2))
        }
      }else if (model == 4){
        for (i in 1:n){
          y[i] <- B0 + B1*X[i,2] + k*(X[i,2]^2 + X[i,2]^3 + X[i,2]^4) +
                    rnorm(1, mean = 0, sd = sqrt(sigma2))
        }
      }

      #Estimate Beta for each simulated dataset
      BetaN[j,] <- solve(t(X)%*%X)%*%t(X)%*%y
    }

    #Compute MSE estimate
    BiasB <- colMeans(BetaN) - c(1,1)
    VarB <- c(var(BetaN[,1]),var(BetaN[,2]))
  }

  #####QUADRATIC AND MLR MODEL
  if (model == 2|model == 3|model == 5|model == 6){
    B2 <- 1
    BetaN <- matrix(0,N,3)

    if (model == 2|model == 5){
      X <- cbind(X,X[,2]^2)
    }

    for (j in 1:N){
      #Simulate residuals and y values
      y <- vector(mode="double",length = n)
      if (model == 2|model == 3){
        for (i in 1:n){
          #Note that this works for both quadratic model and MLR model
          y[i] <- B0 + B1*X[i,2] + B2*X[i,3] + k*X[i,2]*X[i,3] +
                    rnorm(1, mean = 0, sd = sqrt(sigma2))
        }
```

```
    }else if (model == 5){
      for (i in 1:n){
        y[i] <- 1 + X[i,2] + X[i,2]^2 + k*(X[i,2]^3 + X[i,2]^4 + X[i,2]^5) +
                  rnorm(1,mean=0,sd=sigma2)
      }
    }else if (model == 6){
      for (i in 1:n){
        y[i] <- 1 + X[i,2] + X[i,3] + k*(X[i,2]*X[i,3] + X[i,2]^2 + X[i,3]^2) +
                  rnorm(1,mean=0,sd=sigma2)
      }
    }

    #Estimate Beta for each simulated dataset
    BetaN[j,] <- solve(t(X)%*%X)%*%t(X)%*%y
  }

  #Compute bias and variances of beta
  BiasB <- colMeans(BetaN) - c(1,1,1)
  VarB <- c(var(BetaN[,1]),var(BetaN[,2]),var(BetaN[,3]))
}


####Compute Cov and MSE and determinants and output list of values
CovB <- sigma2*solve(t(X)%*%X)
#CovB <- cov(BetaN)
detCovB <- det(CovB)
MSEBhat <- CovB + BiasB%*%t(BiasB)
detMSEBhat <- det(MSEBhat)


output <- list(detMSEBhat,detCovB,BiasB,VarB)
return(output)
}
```

kL2.R

```
    #Function kL2

#Function used to determine k values for model 4,5,6 (linear, quadratic, MLR where
#there are 3 missing true parameters)


kL2 <- function(n,sigma2vector,maxk,model){

  kresults <- vector(mode="double",length = length(sigma2vector))

  for (i in 1:length(sigma2vector)){
    sigma2 <- sigma2vector[i]
    print(sigma2)
    #eta2 - boundary condition
```

```
eta2 <- sigma2/n

if (model == 1|model == 4){
  for (k in seq(maxk,0,by=-0.00001)){

    Betahat <- vector(mode="double",length = 2)
    Betahat[1] <- 1 + 2*k*(0.5^3/3 + 0.5^5/5)
    Betahat[2] <- 1 + 3*k*0.5^2/5

    #integrate until integral is less than eta2
    integral <- integrate(function(x){
      (1 + x + k*(x^2 + x^3 + x^4) - Betahat[1] - Betahat[2]*x)^2
    },lower = -0.5,upper = 0.5)[[1]]

    val <- (1 + 0.75 + k*(0.75^2 + 0.75^3 + 0.75^4) - Betahat[1] - Betahat[2]*0.75)^2

    if (integral < eta2 && val < eta2){
      break
    }
  }
}else if (model == 2|model == 5){
  for (k in seq(maxk,0,by=-0.00001)){

    Betahat <- vector(mode = "double",length = 3)
    Betahat[1] <- -0.5^2/3*(1+k*0.5^2*(1/5 - 3/7)/(1/3 - 3/5)) + 1 + 0.5^2/3 + k*0.5^4/5
    Betahat[2] <- 1 + 3*k*(0.5^2/5 + 0.5^4/7)
    Betahat[3] <- 1 + k*0.5^2*(1/5 - 3/7)/(1/3 - 3/5)

    #integrate until integral is less than eta2
    integral <- integrate(function(x){
      (1 + x + x^2 + k*(x^3 + x^4 + x^5) - Betahat[1] - Betahat[2]*x - Betahat[3]*x^2)^2
    },lower = -0.5,upper = 0.5)[[1]]

    if (integral < eta2){
      break
    }
  }
}else if (model == 3|model == 6){
  for (k in seq(maxk,0,by=-0.00001)){

    Betahat <- vector(mode= "double",length = 3)
    Betahat[1] <- 1 + 2*k*0.5^2/3
    Betahat[2] <- 1
    Betahat[3] <- 1

    #integrate until integral is less than eta2
    integral <- integrate(function(y) {
      sapply(y,function(y){
        integrate(function(x){
          sapply(x,function(x){
```

```
              (1 + x + y + k*(x*y + x^2 + y^2) - Betahat[1] - Betahat[2]*x - Betahat[3]*y)^2
            })
          },-0.5,0.5)$value
        })
      },-0.5,0.5)$value

      if (integral < eta2){
        break
      }
    }
  }

  kresults[i] <- round(k,4)
 }

 return(kresults)
}
```

Extrapolation Codes

RunEP.R

```
    #Source code to run Extrapolation.R for the different models and x0's

#Values of n for each scenario:
#Model 1: x0 = 0.75 -> n = 30,   x0 = 2.5 -> n = 30
#Model 2: x0 = 0.75 -> n = 28,   x0 = 2.5 -> n = 49

library(xtable)
setwd('C:/Users/dburm/Desktop/School/MATH 4F90/')
source('EPfunctions.R')

n <- 30
N <- 50000
sigma2vector <- c(0.01,0.05,0.1,0.2,0.5)

#MODEL 1
model <- 1
x0 <- 0.75
source('Extrapolation.R',echo=T)

x0 <- 2.5
source('Extrapolation.R',echo=T)

#MODEL 2
model <- 2
n <- 28
x0 <- 0.75
source('Extrapolation.R',echo=T)
```

```
n <- 49
x0 <- 2.5
source('Extrapolation.R',echo=T)
```

Extrapolation.R

```
    #***For every function except kL2.R, for model 2 k3 and k4 still use variable names
    k2 and k3 respectively
#(Or k3vector and k4vector have variable names k2vector and k3vector respectively) for
simplicity of code




#Determine k and optimal p
results <- matrix(NA,nrow = length(sigma2vector),ncol = 6+model)
if (model == 1){
  colnames(results) <- c("sigma2","sigma2/n","beta0","beta1","k2","k3","p")
}else if (model == 2){
  colnames(results) <- c("sigma2","sigma2/n","beta0","beta1","beta2","k3","k4","p")
}
results[,1] <- sigma2vector
results[,2] <- sigma2vector/n
if (model == 1){
  results[,3:6] <- kL2(x0,model)
}else if (model == 2){
  results[,3:7] <- kL2(x0,model)
}
results[,6+model] <- Optp(n,N=100,x0,sigma2vector,k2vector = results[,4+model],
                          k3vector = results[,5+model],model = model)
#results[,6+model] <- c(0.05,0.05,0.05,0.05,0.05)


#Classical (HL) design results table
X <- HLdesign(n,x0,model = model)
resultsHL <- resultstable(X,n,N,x0,kresults = results,Clust=F,model = model)

#WX design (Huber's implemented)
X <- WXdesign(n,x0,m = n,model = model)
resultsWX <- resultstable(X,n,N,x0,kresults = results,Clust=F,model = model)

#Clustered design results table
resultsCl <- resultstable(X=F,n,N,x0,kresults = results,Clust=T,model = model)

#Uniform design results table
X <- cbind(rep(1,n),seq(-0.5,0.5,by=1/(n-1)))
if (model == 2){
  X <- cbind(X,X[,2]^2)
}
```

```
resultsU <- resultstable(X,n,N,x0,kresults = results,Clust=F,model = model)

#analysis table
comparisontable <- matrix(NA,nrow = length(sigma2vector)*5,ncol = 4)
colnames(comparisontable) <- c("Bias","Var","MSE","RE")
for (i in 1:length(sigma2vector)){
  comparisontable[5*(i-1)+1, 1:4] <- c(results[i,c(1,4+model,5+model,6+model)]) #sigma,k2,k3,p
  comparisontable[5*(i-1)+2, 1:4] <- c(resultsCl[i,c(4,5,6)],NA)
  comparisontable[5*(i-1)+3, 1:4] <- c(resultsHL[i,c(4,5,6)],resultsHL[i,6]/resultsCl[i,6])
  comparisontable[5*(i-1)+4, 1:4] <- c(resultsWX[i,c(4,5,6)],resultsWX[i,6]/resultsCl[i,6])
  comparisontable[5*(i-1)+5, 1:4] <- c(resultsU[i,c(4,5,6)],resultsU[i,6]/resultsCl[i,6])
}



#Save rds file
EPresults <- list(results = results,
                        resultsHL = resultsHL,
                        resultsCl = resultsCl,
                        resultsWX = resultsWX,
                        resultsU = resultsU,
                        comparisontable = comparisontable)
saveRDS(EPresults,file = paste("model",model,"EPx0_",x0,"results.rds",sep=""))
```

EPfunctions.R

```
    #All functions used in Extrapolation simulation

#####Calculate MSE's and bias, variance for designs
MSEextrapolation <- function(n,N,X,x0,sigma2,k2,k3,model){
  if (model == 1){
    f <- function(x,k2,k3){1 + x + k2*x^2 + k3*x^3}
  }else if (model == 2){
    f <- function(x,k2,k3){1 + x + x^2 + k2*x^3 + k3*x^4}
  }

  #Vector for N extrapolations at x0
  yhat <- vector(mode = "double",length = N)
  if (model == 1){
    betahat <- matrix(NA,ncol = N,nrow = 2)
  }else if (model == 2){
    betahat <- matrix(NA,ncol = N,nrow = 3)
  }

  for (j in 1:N){
    #Sample data
    y <- vector(mode="double",length = n)
    for (i in 1:n){
      y[i] <- f(X[i,2],k2,k3) + rnorm(1,0,sd = sigma2)
```

```
    }
    #Estimate parameters
    Beta <- solve(t(X)%*%X)%*%t(X)%*%y
    #Estimate extrapolation
    if (model == 1){
      yhat[j] <- Beta[1] + Beta[2]*x0
    }else if (model == 2){
      yhat[j] <- Beta[1] + Beta[2]*x0 + Beta[3]*x0^2
    }
    betahat[,j] <- Beta
  }

  #Extrapolation Bias, Variance and MSE
  if (model == 1){
    #Asymptotic formulas
    bias <- c(1,x0)%*%solve(t(X)%*%X)%*%t(X)%*%f(X[,2],k2,k3) - f(x0,k2,k3)
    variance <- c(1,x0)%*%solve(t(X)%*%X)%*%c(1,x0)*sigma2
    # #Simulated formulas
    # bias <- mean(yhat) - f(x0,k2,k3)
    # variance <- var(yhat)
  }else if (model == 2){
    #Asymptotic formulas
    bias <- c(1,x0,x0^2)%*%solve(t(X)%*%X)%*%t(X)%*%f(X[,2],k2,k3) - f(x0,k2,k3)
    variance <- c(1,x0,x0^2)%*%solve(t(X)%*%X)%*%c(1,x0,x0^2)*sigma2
    # #Simulated formulas
    # bias <- mean(yhat) - f(x0,k2,k3)
    # variance <- var(yhat)
  }
  MSE <- variance + bias^2

  betahat <- rowMeans(betahat)

  return(c(bias,variance,MSE,betahat))
}


#####Creates design matrix according to HL design
HLdesign <- function(n,x0,model){
  if (model == 1){
    x02 <- 2*x0
    w <- c((x02-1)/(2*x02),(x02+1)/(2*x02))
    xi <- c(rep(-0.5,n*w[1]),rep(0.5,n*w[2]))
  }else if (model == 2){
    x02 <- 2*x0
    w <- c(x02*(x02-1)/(2*(2*x02^2 - 1)),
           (x02-1)*(x02+1)/(2*x02^2 - 1),
           x02*(x02+1)/(2*(2*x02^2 - 1)))
    xi <- c(rep(-0.5,n*w[1]),rep(0,n*w[2]),rep(0.5,n*w[3]))
  }
  if (model == 1){
```

```
    X <- cbind(rep(1,n),xi)
  }else if (model == 2){
    X <- cbind(rep(1,n),xi,xi^2)
  }

  return(X)
}

#####Creates design matrix according to WX design
WXdesign <- function(n,x0,m,model){
  if (model == 1){
    if (x0 == 0.75){
      a1 <- 14.06
      a2 <- 3.18
      a3 <- 11.23
      a4 <- 1
      a5 <- -16.52
      a6 <- 0.0032
    }else if (x0 == 2.5){
      a1 <- 148.64
      a2 <- 9.73
      a3 <- 80.38
      a4 <- 1
      a5 <- -525.93
      a6 <- 0.000122
    }
    f <- function(x){
      pmax(((a1*x*2 + a2)*(a3*x*2 + a4) + a5),0)/((a3*x*2 + a4)^2 + a6*(a3*x*2 + a4)^4)
    }
    totint <- integrate(f,-0.5,0.5)$value
    f <- function(x){
      (pmax(((a1*x*2 + a2)*(a3*x*2 + a4) + a5),0)/((a3*x*2 + a4)^2 + a6*(a3*x*2 + a4)^4))/totint
    }
  }else if (model == 2){
    if (x0 == 0.75){
      b0 <- 1.23
      b1 <- -0.858
      b2 <- -4.10
      a0 <- 1
      a1 <- -0.0710
      a2 <- -2.17
      c <- -0.396
      d <- 0.627
    }else if (x0 == 2.5){
      b0 <- 1.69
      b1 <- -0.255
      b2 <- -5.01
      a0 <- 1
      a1 <- -0.0079
      a2 <- -2.11
```

```
    c <- -0.482
    d <- 0.949
  }
  f <- function(x){
    pmax((a0 + a1*x*2 + a2*x^2*4)*(b0 + b1*x*2 + b2*x^2*4) + c,0)/
          ((a0 + a1*x*2 + a2*x^2*4)^2 + d*(a0 + a1*x*2 + a2*x^2*4)^4)
  }
  totint <- integrate(f,-0.5,0.5)$value
  f <- function(x){
    pmax((a0 + a1*x*2 + a2*x^2*4)*(b0 + b1*x*2 + b2*x^2*4) + c,0)/
          ((a0 + a1*x*2 + a2*x^2*4)^2 + d*(a0 + a1*x*2 + a2*x^2*4)^4)/totint
  }
}

#integrate until Huber's equation is satisfied
h <- 0.0001
xj <- vector(mode="double",length = m)

for(j in 1:m){
  i <- -0.5
  val <- 0
  while(val < (j - 0.5)/m){
    val <- integrate(f,-0.5,i)$value
    i <- i + h
  }
  xj[j] <- i
}
if (model == 1){
  X <- cbind(rep(1,n),xj)
}else if (model == 2){
  X <- cbind(rep(1,n),xj,xj^2)
}

return(X)
}

#####Clustered Design for extrapolation
Cldesign <- function(n,x0,p,model){
  if (model == 1){
    x02 <- 2*x0
    w <- c((x02-1)/(2*x02),(x02+1)/(2*x02))
    xi <- c(seq(-0.5,-0.5+p/2,by = (p/2)/(n*w[1] - 1)),
            seq(0.5-p/2,0.5, by = (p/2)/(n*w[2] - 1)))
  }else if (model == 2){
    x02 <- 2*x0
    w <- c(x02*(x02-1)/(2*(2*x02^2 - 1)),
           (x02-1)*(x02+1)/(2*x02^2 - 1),
           x02*(x02+1)/(2*(2*x02^2 - 1)))
    xi <- c(seq(-0.5,-0.5+p/3,by = (p/3)/(n*w[1] - 1)),
            seq(-p/6,p/6,by = (p/3)/(n*w[2] - 1)),
```

```
           seq(0.5-p/3,0.5,by = (p/3)/(n*w[3] - 1)))
  }
  if (model == 1){
    X <- cbind(rep(1,n),xi)
  }else if (model == 2){
    X <- cbind(rep(1,n),xi,xi^2)
  }


  return(X)
}

#####Find optimal p for extrapolation
Optp <- function(n,N,x0,sigma2vector,k2vector,k3vector,model){
  prange <- seq(0.05,0.95,by=0.01)
  opt <- vector(mode="double",length = length(sigma2vector))
  #Matrix for storing estimated MSE for each p for each sigma^2
  pMSE <- matrix(0,nrow = length(prange), ncol = length(sigma2vector))
  colnames(pMSE) <- c("0.01","0.05","0.1","0.2","0.5")

  for (joe in 1:10){
    print(joe)
    for (j in 1:length(prange)){
      p <- prange[j]
      X <- Cldesign(n,x0,p,model)

      #Estimate det(MSE) for each sigma^2 for this p
      for (i in 1:length(sigma2vector)){
        pMSE[j,i] <- pMSE[j,i] + MSEextrapolation(n,N,X,x0,sigma2vector[i],k2vector[i],
                                                  k3vector[i],model = model)[[3]]
        #pMSE[j,i] <- pMSE[j,i] + MSEextrapolation(n,N,X,x0,sigma2vector[i],k2vector[i],
                                                  k3vector[i],model = model)[[1]]
      }
    }
  }
  pMSE <- pMSE/10

  #Plots
  for(i in 1:length(sigma2vector)){
    #Code for printing plots
    if (model == 1){
      plot(prange,pMSE[,i],xlab = "p",ylab = "MSE(yhat(x0))", pch = 19,col = "gray45",
           main = paste("Model ",model,".  sigma^2 =",sigma2vector[i],"\nk2 =",k2vector[i],
                        ", k3 =",k3vector[i],sep = ""))
    }else if (model == 2){
      plot(prange,pMSE[,i],xlab = "p",ylab = "MSE(yhat(x0))", pch = 19,col = "gray45",
           main = paste("Model ",model,".  sigma^2 =",sigma2vector[i],"\nk3 =",k2vector[i],
                        ", k4 =",k3vector[i],sep = ""))
    }
```

```
    fhat <- ksmooth(prange,pMSE[,i],kernel = "normal", bandwidth = 0.15,
                    range.x = range(prange),n.points = 100, x.points = prange)

    #Code for saving plots
    lines(fhat)
    dev.copy(png,paste("model",model,"s",sigma2vector[i],"x0_",x0,"EP.png",sep=""))
    dev.off()

    opt[i] <- fhat$x[which.min(fhat$y)]
  }
  return(opt)
}

#####Results tables function
resultstable <- function(X,n,N,x0,kresults,Clust,model){
  sigma2vector <- kresults[,1]
  k2vector <- kresults[,4+model]
  k3vector <- kresults[,5+model]
  pvector <- kresults[,6+model]

  results <- matrix(NA,nrow = length(sigma2vector),ncol = 6)
  if (model == 1){
    colnames(results) <- c("Sigma^2","k2","k3","EBias","EVariance","EMSE")
  }else if (model == 2){
    colnames(results) <- c("Sigma^2","k3","k4","EBias","EVariance","EMSE")
  }
  results[,1] <- sigma2vector
  results[,2] <- k2vector
  results[,3] <- k3vector
  for (i in 1:length(sigma2vector)){
    if (Clust == T){
      p <- pvector[i]
      X <- Cldesign(n,x0,p,model)
    }
    design <- MSEextrapolation(n,N,X,x0,sigma2 = sigma2vector[i],k2 = k2vector[i],
                               k3 = k3vector[i],model = model)
    results[i,4] <- design[[1]] #EBias
    results[i,5] <- design[[2]] #EVariance
    results[i,6] <- design[[3]] #EMSE
  }

  if (Clust == T){
    results <- cbind(results,pvector)
    colnames(results)[7] <- "p"
  }
  return(results)
}


#####The following results were obtained in Maple
```

```
kL2 <- function(x0,model,sigma2vector,n,maxk){
  values <- matrix(NA,nrow = 5,ncol = 3+model)

  if (model == 1){
    if (x0 == 0.75){     #Second extrapolation bias condition
      values[,1] <- c(1.0198,1.0443,1.0626,1.0885,1.1399) #beta0
      values[,2] <- c(0.9645,0.9206,0.8877,0.8412,0.7489) #beta1
      values[,3] <- c(0.2375,0.5310,0.7510,1.0621,1.6793) #k2
      values[,4] <- c(-0.2367,-0.5293,-0.7485,-1.0586,-1.6738) #k3
    }else if (x0 == 2.5){
      values[,1] <- c(1.0021,1.0040,1.0066,1.0093,1.0140) #beta0
      values[,2] <- c(0.9982,0.9963,0.9943,0.9924,0.9874) #beta1
      values[,3] <- c(0.025,0.048,0.079,0.111,0.168) #k2
      values[,4] <- c(-0.012,-0.025,-0.038,-0.051,-0.084) #k3
    }


  }else if (model == 2){
    if (x0 == 0.75){     #Second extrapolation bias condition
      values[,1] <- c(0.9901,0.9778,0.9686,0.9556,0.9298) #beta0
      values[,2] <- c(0.8674,0.7034,0.5806,0.4069,0.0622) #beta1
      values[,3] <- c(1.3972,1.8883,2.2562,2.7765,3.8089) #beta2
      values[,4] <- c(-0.8842,-1.9771,-2.7961,-3.9543,-6.2522) #k3
      values[,5] <- c(1.8538,4.1452,5.8622,8.2904,13.1083) #k4
    }else if (x0 == 2.5){
      values[,1] <- c(1.0000,1.0000,1.0001,1.0001,1.0002) #beta0
      values[,2] <- c(1.0005,1.0033,1.0053,1.0080,1.0122) #beta1
      values[,3] <- c(0.9996,0.9981,0.9964,0.9951,0.9916) #beta2
      values[,4] <- c(0.003,0.022,0.035,0.053,0.081) #k3
      values[,5] <- c(-0.002,-0.009,-0.017,-0.023,-0.039) #k4
    }

  }

  return(values)
}
```

2.5kvalsmodel1.R

```
    rm(list=ls())
cat('\f')

sigma2vector <- c(0.01,0.05,0.1,0.2,0.5)
x0 <- 2.5
n <- 30

kresults <- matrix(NA,nrow = 8,ncol = 5)
colnames(kresults) <- c("0.01","0.05","0.1","0.2","0.5")
rownames(kresults) <- c("beta0","beta1","k2","k3","er1","er2","er1+er2","sigma2/n")
```

```
kmin <- 0.001
kmax <- 2
kby <- kmin


######er1 only
for (j in 1:length(sigma2vector)){
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = length(seq(-kmax,kmax,by = kby)),ncol = 5)
  colnames(table) <- c("k2","k3","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,2] <- sort(seq(-kmax,kmax,by = kby))

  for (i in 1:nrow(table)){
    #Ignore cases with imaginary solutions
    if (is.na( (5.353955978*10^(-10)*(-3.845*10^9*table[i,2]*n +
        sqrt(-4.291584153*10^17*table[i,2]^2*n^2 + 1.867777778*10^17*n*sigma^2)))/n) == T){
      table[i,1] <- NA
      table[i,3] <- NA
      table[i,4] <- NA
      table[i,5] <- NA
    }else{
      #k2 values
      table[i,1] <- (5.353955978*10^(-10)*(-3.845*10^9*table[i,2]*n +
                sqrt(-4.291584153*10^17*table[i,2]^2*n^2 + 1.867777778*10^17*n*sigma^2)))/n
      #define function given k2,k3
      beta0 <- 1 + table[i,1]/12
      beta1 <- 1 + 3*table[i,2]/20
      f <- function(x){(1 + x + table[i,1]*x^2 + table[i,2]*x^3 - beta0 - beta1*x)^2}

      #compute errors and sum of errors
      table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
      table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
      table[i,5] <- table[i,3] + table[i,4]
    }
  }
  table2 <- table[is.na(table[,1])==F,]
  table2 <- table2[table2[,3]>=0,]
  if (nrow(table2) == 0){
    print(paste("sigma2 = ",sigma^2," - no valid solutions"))
  }else{
    kresults[3,j] <- table2[which.min(table2[,3]),1]
    kresults[4,j] <- table2[which.min(table2[,3]),2]
    kresults[1,j] <- 1 + kresults[3,j]/12
    kresults[2,j] <- 1 + 3*kresults[4,j]/20

    #Errors
```

```
    f <- function(x){(1 + x + kresults[3,j]*x^2 + kresults[4,j]*x^3 - kresults[1,j] -
                        kresults[2,j]*x)^2}
    kresults[5,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    kresults[6,j] <- sigma^2/n - integrate(f,0.5,x0)$value
    kresults[7,j] <- kresults[5,j] + kresults[6,j]

    kresults[8,j] <- sigma^2/n
  }
}



######er2 only
for (j in 1:length(sigma2vector)){
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = length(seq(-kmax,kmax,by = kby)),ncol = 5)
  colnames(table) <- c("k2","k3","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,2] <- sort(seq(-kmax,kmax,by = kby))

  for (i in 1:nrow(table)){
    #Ignore cases with imaginary solutions
    if (is.na(3*sqrt(-35*n*(n*table[i,2]^2 - 2800*sigma^2))/(70*n)) == T){
      table[i,1] <- NA
      table[i,3] <- NA
      table[i,4] <- NA
      table[i,5] <- NA
    }else{
      #k2 values
      table[i,1] <- 3*sqrt(-35*n*(n*table[i,2]^2 - 2800*sigma^2))/(70*n)
      #define function given k2,k3
      beta0 <- 1 + table[i,1]/12
      beta1 <- 1 + 3*table[i,2]/20
      f <- function(x){(1 + x + table[i,1]*x^2 + table[i,2]*x^3 - beta0 - beta1*x)^2}

      #compute errors and sum of errors
      table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
      table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
      table[i,5] <- table[i,3] + table[i,4]
    }
  }

  table2 <- table[is.na(table[,1])==F,]
  table2 <- table2[table2[,4]>=0,]

  if (nrow(table2) == 0){
    print(paste("sigma2 = ",sigma^2," - no valid solutions"))
  }else{
```

```
    kresults[3,j] <- table2[which.min(table2[,4]),1]
    kresults[4,j] <- table2[which.min(table2[,4]),2]
    kresults[1,j] <- 1 + kresults[3,j]/12
    kresults[2,j] <- 1 + 3*kresults[4,j]/20

    #Errors
    f <- function(x){(1 + x + kresults[3,j]*x^2 + kresults[4,j]*x^3 - kresults[1,j] -
                        kresults[2,j]*x)^2}
    kresults[5,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    kresults[6,j] <- sigma^2/n - integrate(f,0.5,x0)$value
    kresults[7,j] <- kresults[5,j] + kresults[6,j]

    kresults[8,j] <- sigma^2/n
  }
}



######er1 + er2
for (j in 1:length(sigma2vector)){
  ktot <- length(seq(kmin,kmax,by = kby))
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = ktot^2,ncol = 5)
  colnames(table) <- c("k2","k3","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,1:2] <- cbind(sort(rep(seq(kmin,kmax,by = kby),ktot)),
                       rep(seq(-kmin,-kmax,by = -kby),ktot))
  #compute errors for each combination of k2,k3
  for (i in 1:nrow(table)){
    #define function given k2,k3
    beta0 <- 1 + table[i,1]/12
    beta1 <- 1 + 3*table[i,2]/20
    f <- function(x){(1 + x + table[i,1]*x^2 + table[i,2]*x^3 - beta0 - beta1*x)^2}

    #compute errors and sum of errors
    table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
    table[i,5] <- table[i,3] + table[i,4]
  }

  #remove negative errors
  table2 <- table[table[,3]>=0,]
  table2 <- table2[table2[,4]>=0,]

  #record results for sigma
  kresults[3,j] <- table2[which.min(table2[,5]),1]
  kresults[4,j] <- table2[which.min(table2[,5]),2]
  kresults[1,j] <- 1 + kresults[3,j]/12
```

```
  kresults[2,j] <- 1 + 3*kresults[4,j]/20

  #Errors
  f <- function(x){(1 + x + kresults[3,j]*x^2 + kresults[4,j]*x^3 - kresults[1,j] -
                    kresults[2,j]*x)^2}
  kresults[5,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
  kresults[6,j] <- sigma^2/n - integrate(f,0.5,x0)$value
  kresults[7,j] <- kresults[5,j] + kresults[6,j]

  kresults[8,j] <- sigma^2/n
}


kresults <- round(kresults,digits = 6)
```

2.5kvalsmodel2.R

```
    rm(list=ls())
cat('\f')

sigma2vector <- c(0.01,0.05,0.1,0.2,0.5)
x0 <- 2.5
n <- 49

kresults <- matrix(NA,nrow = 9,ncol = 5)
colnames(kresults) <- c("0.01","0.05","0.1","0.2","0.5")
rownames(kresults) <- c("beta0","beta1","beta2","k3","k4","er1","er2","er1+er2","sigma2/n")

kmin <- 0.001
kmax <- 3
kby <- kmin


######er1 only
for (j in 1:length(sigma2vector)){
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = length(seq(-kmax,kmax,by = kby)),ncol = 5)
  colnames(table) <- c("k3","k4","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,2] <- sort(seq(-kmax,kmax,by = kby))

  for (i in 1:nrow(table)){
    #Ignore cases with imaginary solutions
    if (is.na( (1.227736317*10^(-10))*(-1.762757143*10^10*table[i,2]*n +
        sqrt(-4.956216559*10^18*table[i,2]^2*n^2 + 8.145071429*10^17*n*sigma^2)))/n) == T){
      table[i,1] <- NA
      table[i,3] <- NA
      table[i,4] <- NA
```

```
      table[i,5] <- NA
    }else{
      #k3 values
      table[i,1] <- (1.227736317*10^(-10)*(-1.762757143*10^10*table[i,2]*n +
            sqrt(-4.956216559*10^18*table[i,2]^2*n^2 + 8.145071429*10^17*n*sigma^2)))/n
      #define function given k3,k4
      beta2 <- 1 + 3*table[i,2]/14
      beta0 <- 13/12 + table[i,2]/80 - beta2/12
      beta1 <- 1 + 3*table[i,1]/20


      f <- function(x){(1 + x + x^2 + table[i,1]*x^3 + table[i,2]*x^4 - beta0 - beta1*x -
                        beta2*x^2)^2}

      #compute errors and sum of errors
      table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
      table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
      table[i,5] <- table[i,3] + table[i,4]
    }
  }
  table2 <- table[is.na(table[,1])==F,]
  table2 <- table2[table2[,3]>=0,]
  if (nrow(table2) == 0){
    print(paste("sigma2 = ",sigma^2," - no valid solutions"))
  }else{
    kresults[4,j] <- table2[which.min(table2[,3]),1] #k3
    kresults[5,j] <- table2[which.min(table2[,3]),2] #k4
    kresults[3,j] <- 1 + 3*kresults[5,j]/14 #beta2
    kresults[1,j] <- 13/12 + kresults[5,j]/80 - kresults[3,j]/12 #beta0
    kresults[2,j] <- 1 + 3*kresults[4,j]/20 #beta1

    #Errors
    f <- function(x){(1 + x + x^2 + kresults[4,j]*x^3 + kresults[5,j]*x^4 - kresults[1,j] -
                        kresults[2,j]*x - kresults[3,j]*x^2)^2}
    kresults[6,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    kresults[7,j] <- sigma^2/n - integrate(f,0.5,x0)$value
    kresults[8,j] <- kresults[6,j] + kresults[7,j]

    kresults[9,j] <- sigma^2/n
  }
}


######er2 only
for (j in 1:length(sigma2vector)){
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = length(seq(-kmax,kmax,by = kby)),ncol = 5)
```

```
  colnames(table) <- c("k3","k4","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,2] <- sort(seq(-kmax,kmax,by = kby))

  for (i in 1:nrow(table)){
    #Ignore cases with imaginary solutions
    if (is.na( 2*sqrt(-7*n*(table[i,2]^2*n - 44100*sigma^2))/(21*n)) == T){
      table[i,1] <- NA
      table[i,3] <- NA
      table[i,4] <- NA
      table[i,5] <- NA
    }else{
      #k3 values
      table[i,1] <- 2*sqrt(-7*n*(table[i,2]^2*n - 44100*sigma^2))/(21*n)
      #define function given k3,k4
      beta2 <- 1 + 3*table[i,2]/14
      beta0 <- 13/12 + table[i,2]/80 - beta2/12
      beta1 <- 1 + 3*table[i,1]/20


      f <- function(x){(1 + x + x^2 + table[i,1]*x^3 + table[i,2]*x^4 - beta0 - beta1*x -
                        beta2*x^2)^2}

      #compute errors and sum of errors
      table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
      table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
      table[i,5] <- table[i,3] + table[i,4]
    }
  }
  table2 <- table[is.na(table[,1])==F,]
  table2 <- table2[table2[,4]>=0,]
  if (nrow(table2) == 0){
    print(paste("sigma2 = ",sigma^2," - no valid solutions"))
  }else{
    kresults[4,j] <- table2[which.min(table2[,4]),1] #k3
    kresults[5,j] <- table2[which.min(table2[,4]),2] #k4
    kresults[3,j] <- 1 + 3*kresults[5,j]/14 #beta2
    kresults[1,j] <- 13/12 + kresults[5,j]/80 - kresults[3,j]/12 #beta0
    kresults[2,j] <- 1 + 3*kresults[4,j]/20 #beta1

    #Errors
    f <- function(x){(1 + x + x^2 + kresults[4,j]*x^3 + kresults[5,j]*x^4 - kresults[1,j] -
                        kresults[2,j]*x - kresults[3,j]*x^2)^2}
    kresults[6,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    kresults[7,j] <- sigma^2/n - integrate(f,0.5,x0)$value
    kresults[8,j] <- kresults[6,j] + kresults[7,j]

    kresults[9,j] <- sigma^2/n
  }
}
```

```
######er1 + er2
for (j in 1:length(sigma2vector)){
  ktot <- length(seq(kmin,kmax,by = kby))
  sigma <- sqrt(sigma2vector[j])
  print(sigma2vector[j])

  #table for error results for each k2,k3
  table <- matrix(NA,nrow = ktot^2,ncol = 5)
  colnames(table) <- c("k3","k4","er1=eta2-intS","er2=eta2-intExtrap","er1+er2")
  table[,1:2] <- cbind(sort(rep(seq(kmin,kmax,by = kby),ktot)),
                       rep(seq(-kmin,-kmax,by = -kby),ktot))

  for (i in 1:nrow(table)){

    #define function given k3,k4
    beta2 <- 1 + 3*table[i,2]/14
    beta0 <- 13/12 + table[i,2]/80 - beta2/12
    beta1 <- 1 + 3*table[i,1]/20
    f <- function(x){(1 + x + x^2 + table[i,1]*x^3 + table[i,2]*x^4 - beta0 - beta1*x -
                       beta2*x^2)^2}

    #compute errors and sum of errors
    table[i,3] <- sigma^2/n - integrate(f,-0.5,0.5)$value
    table[i,4] <- sigma^2/n - integrate(f,0.5,x0)$value
    table[i,5] <- table[i,3] + table[i,4]
  }

  #remove negative errors
  table2 <- table[table[,3]>=0,]
  table2 <- table2[table2[,4]>=0,]

  kresults[4,j] <- table2[which.min(table2[,5]),1] #k3
  kresults[5,j] <- table2[which.min(table2[,5]),2] #k4
  kresults[3,j] <- 1 + 3*kresults[5,j]/14 #beta2
  kresults[1,j] <- 13/12 + kresults[5,j]/80 - kresults[3,j]/12 #beta0
  kresults[2,j] <- 1 + 3*kresults[4,j]/20 #beta1

  #Errors
  f <- function(x){(1 + x + x^2 + kresults[4,j]*x^3 + kresults[5,j]*x^4 - kresults[1,j] -
                     kresults[2,j]*x - kresults[3,j]*x^2)^2}
  kresults[6,j] <- sigma^2/n - integrate(f,-0.5,0.5)$value
  kresults[7,j] <- sigma^2/n - integrate(f,0.5,x0)$value
  kresults[8,j] <- kresults[6,j] + kresults[7,j]

  kresults[9,j] <- sigma^2/n
}
```

```
kresults <- round(kresults,digits = 6)
```

CIrun.R

```
    #Source code for running CIcomparison.R for all models and x0's

library(xtable)
setwd('C:/Users/dburm/Desktop/School/MATH 4F90/')
source('EPfunctions.R')
source('CIfunction.R')

N <- 50000
sigma2vector <- c(0.01,0.05,0.1,0.2,0.5)


model <- 1
n <- 30
x0 <- 0.75
rdsfile <- readRDS('rds files/model1EPx0_0.75results.rds')
results <- rdsfile$results
source('CIcomparison.R',echo=T)


x0 <- 2.5
rdsfile <- readRDS('rds files/model1EPx0_2.5results.rds')
results <- rdsfile$results
source('CIcomparison.R',echo=T)

model <- 2
n <- 28
x0 <- 0.75
rdsfile <- readRDS('rds files/model2EPx0_0.75results.rds')
results <- rdsfile$results
source('CIcomparison.R',echo=T)

n <- 49
x0 <- 2.5
rdsfile <- readRDS('rds files/model2EPx0_2.5results.rds')
results <- rdsfile$results
source('CIcomparison.R',echo=T)
```

CIcomparison.R

```
    #Code for Relative Bias, Coverage Percentage and avg length of each conf. interval
#Computes all above values of interest and exports results into a latex table


comptable <- matrix(NA,nrow = 5*length(sigma2vector),ncol = 7)
```

```
colnames(comptable) <- c("rel bias (%)","ECI CP","ACI CP","PI CP","ECI avg length",
                         "ACI avg length","PI avg length")
rownames(comptable) <- rep(c("","HL","WX","Cl","U"),length(sigma2vector))

comptable2 <- matrix(NA,nrow = 5*length(sigma2vector),ncol = 4)
colnames(comptable2) <- c("True y(x0)","ybar(x0)","SE","ASD")
rownames(comptable2) <- rep(c("","HL","WX","Cl","U"),length(sigma2vector))

for (i in 1:length(sigma2vector)){
  comptable[5*(i-1)+1,1:4] <- c(results[i,c(1,4+model,5+model,6+model)])
  comptable2[5*(i-1)+1,1:4] <- c(results[i,c(1,4+model,5+model,6+model)])
}

#HL design
X <- HLdesign(n,x0,model = model)
for (i in 1:length(sigma2vector)){
  values <- CIfunction(n,N,X,x0,sigma2vector[i],results[i,4+model],results[i,5+model],
                       model = model)
  comptable[5*(i-1)+2,] <- values[1:7]
  comptable2[5*(i-1)+2,] <- values[8:11]
}

#WX design
X <- WXdesign(n,x0,m = n,model = model)
for (i in 1:length(sigma2vector)){
  values <- CIfunction(n,N,X,x0,sigma2vector[i],results[i,4+model],results[i,5+model],
                       model = model)
  comptable[5*(i-1)+3,] <- values[1:7]
  comptable2[5*(i-1)+3,] <- values[8:11]
}

#Clustered Design
for (i in 1:length(sigma2vector)){
  X <- Cldesign(n,x0,results[i,6+model],model=model)
  values <- CIfunction(n,N,X,x0,sigma2vector[i],results[i,4+model],results[i,5+model],
                       model = model)
  comptable[5*(i-1)+4,] <- values[1:7]
  comptable2[5*(i-1)+4,] <- values[8:11]
}

#Uniform Design
X <- cbind(rep(1,n),seq(-0.5,0.5,by=1/(n-1)))
if (model == 2){
  X <- cbind(X,X[,2]^2)
}
for (i in 1:length(sigma2vector)){
  values <- CIfunction(n,N,X,x0,sigma2vector[i],results[i,4+model],results[i,5+model],
                       model = model)
  comptable[5*i,] <- values[1:7]
  comptable2[5*i,] <- values[8:11]
```

```
}


#Save results as RDS
saveitems <- list(comptable,comptable2)
saveRDS(saveitems,file = paste("model",model,"EPx0_",x0,"CI.rds",sep=""))
```

CIfunction.R

```
    #Computes all measurements of interest for extrapolation
#Including: relative bias, ECI, ACI, PI, interval lengths, simulated SE, average ASD

CIfunction <- function(n,N,X,x0,sigma2,k2,k3,model){
  if (model == 1){
    f <- function(x,k2,k3){1 + x + k2*x^2 + k3*x^3}
    z <- c(1,x0)
  }else if (model == 2){
    f <- function(x,k2,k3){1 + x + x^2 + k2*x^3 + k3*x^4}
    z <- c(1,x0,x0^2)
  }

  #Vector for N extrapolations at x0
  yhat <- vector(mode = "double", length = N)
  betahat <- matrix(NA,nrow = model + 1,ncol = N)
  sigma2hat <- vector(mode = "double", length = N)
  for (j in 1:N){
    #Sample data
    y <- vector(mode="double",length = n)
    for (i in 1:n){
      y[i] <- f(X[i,2],k2,k3) + rnorm(1,0,sd = sigma2)
    }
    #Estimate parameters
    Beta <- solve(t(X)%*%X)%*%t(X)%*%y
    #Estimate extrapolation
    if (model == 1){
      yhat[j] <- Beta[1] + Beta[2]*x0
    }else if (model == 2){
      yhat[j] <- Beta[1] + Beta[2]*x0 + Beta[3]*x0^2
    }
    betahat[,j] <- Beta

    sigma2hat[j] <- sum((y - yhat[j])^2)/(n - (model + 1))
  }

  output <- matrix(NA,nrow = 1,ncol = 11)
  colnames(output) <- c("rel bias (%)","ECI CP","ACI CP","PI CP","ECI avg length",
                        "ACI avg length","PI avg length",
                        "True y(x_0)","ybar(x_0)","sim SE","ASD")

  y_x0 <- f(x0,k2,k3)
```

```
###Relative Bias
output[1] <- (mean(yhat) - y_x0)/y_x0*100

###Coverage Percentage and CI lengths
###Empirical CI (ECI)
ECIs <- matrix(NA,nrow = 2,ncol = N)
ECIs[1,] <- yhat + qt(0.05/2, df = n-(model+1),lower.tail = F)*sd(yhat)
ECIs[2,] <- yhat - qt(0.05/2, df = n-(model+1),lower.tail = F)*sd(yhat)
count <- 0
for (j in 1:N){
  if (y_x0 >= ECIs[2,j] && y_x0 <= ECIs[1,j]){
    count <- count + 1
  }
}
output[2] <- count/N*100
output[5] <- mean(ECIs[1,] - ECIs[2,])

###Asymptotic CI (ACI)
ACIs <- matrix(NA,nrow = 2,ncol = N)
ACIs[1,] <- yhat + qt(0.05/2, df = n-(model+1),lower.tail = F)*sqrt(sigma2hat)*
                    c(sqrt(z%*%solve(t(X)%*%X)%*%z))
ACIs[2,] <- yhat - qt(0.05/2, df = n-(model+1),lower.tail = F)*sqrt(sigma2hat)*
                    c(sqrt(z%*%solve(t(X)%*%X)%*%z))
count <- 0
for (j in 1:N){
  if (y_x0 >= ACIs[2,j] && y_x0 <= ACIs[1,j]){
    count <- count + 1
  }
}
output[3] <- count/N*100
output[6] <- mean(ACIs[1,] - ACIs[2,])

###Prediction Interval (PI)
PIs <- matrix(NA,nrow = 2,ncol = N)
PIs[1,] <- yhat + qt(0.05/2, df = n-(model+1),lower.tail = F)*sqrt(sigma2hat)*
                    c(sqrt(z%*%solve(t(X)%*%X)%*%z + 1))
PIs[2,] <- yhat - qt(0.05/2, df = n-(model+1),lower.tail = F)*sqrt(sigma2hat)*
                    c(sqrt(z%*%solve(t(X)%*%X)%*%z + 1))
count <- 0
for (j in 1:N){
  if (y_x0 >= PIs[2,j] && y_x0 <= PIs[1,j]){
    count <- count + 1
  }
}
output[4] <- count/N*100
output[7] <- mean(PIs[1,] - PIs[2,])

output[8] <- y_x0     #True y(x0)
output[9] <- mean(yhat)  #ybar(x0)
output[10] <- sd(yhat)   #simulated SE
```

```
  output[11] <- mean(sqrt(sigma2hat))*c(sqrt(z%*%solve(t(X)%*%X)%*%z))
                                    #average Asymptotic Stand. Dev. (ASD)

  return(output)
}
```