OFF-LINE QUALITY CONTROL BY ROBUST PARAMETER DESIGN

by

JUN YOUNG MIN

B.S., Iowa State University, 2005

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2008

Approved by:

Major Professor
Shie-Shien Yang

Copyright

Abstract

There have been considerable debates over the robust parameter design. As a result, there have been many approaches presented that are suited to the robust parameter design. In my report, I illustrate and present Taguchi's robust parameter design, response surface approach and semi-parameter design.

Considerable attention has been placed on the semi-parameter design. This approach is new technology that was introduced to Picke, Robinson, Birch and Anderson-Cook (2006). The method is a combined parametric and nonparametric technique to improve the estimates of both the mean and the variance of the response.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the faculty in the Department of Statistics for providing a wonderful environment for professional learning. Especially, I would like to express my appreciation to my major professor, Dr. Yang. He has been a great help to this report. I appreciate to the time and effort he spent explaining these concepts and techniques that I have learned in his course and in his office. I appreciate to Dr. John Boyer, James Neill and James Higgins. I would like to acknowledge the following people: Yoonsung Jung, Robert Poulson, Nishantha Samarakoon, Dr. Son and Pastor Yoo.

I am very appreciative of my parents and family for their endless love and support throughout the entirety of my education. Thank you very much to my Lord.

Dedication

To my parents, my family and my Lord!

CHAPTER 1 - Introduction

In the 1970s, AT & T Bell Laboratories, Ford Motor Company, Xerox, DuPont, and many large industrial companies developed high quality products using well planned statistical experiments. There are, however, environmental factors such as temperature, humidity, vibration, and moisture that prevent products from meeting the specified quality that can be attained in the controlled environment of a production plant (Myers et al., 1992). For example, an automobile manufacturing engineer may focus on developing a high-quality tire, finding the right rubber compound composition to produce a tire that lasts at least fifty thousand miles in the factory laboratory. However, if road or weather conditions and driving habits affect the quality of the tire, this expected quality may not be realized uniformly. Although average quality may be near specifications, quality may vary throughout a wide range.

Designing products with quality close to specifications with minimum variability was formalized into a statistical methodology by Taguchi. He called the experimental design that will produce a product with the above characteristics a robust parameter design (Myer et al., 1992; Borror et al., 1999; Robinson et al., 2002). Taguchi called factors intrinsically involved in designing and making the product controllable factors (variables). These factors will directly affect the quality and make up of the product. He called environmental factors that affect or change the quality of the products when they are used outside of the factory the noise factors (Shoemaker et al., 1991; Jones and Box, 1990). He also suggested using a summary statistic, the signal-to-noise ratio, to provide information about the mean and variance of some measurement of quality. Taguchi's signal-to-noise ratio is the performance statistic that he recommended for selecting the best setting of control factors (Pignatiell, 1988; Myer et al., 1992; Borror et al.,

1999). Although Taguchi contributed by developing the robust parameter design, other authors (Shoemaker, Tsui and Wu, 1991; Nair et al., 1992; Borror et al.,1999) have indicated that his approach has some weaknesses: (1) the robust parametric design requires a large of number runs; (2) it uses the signal-to-noise ratio exclusively as the basis for analysis; and (3) it does not consider interaction among controllable factors.

Taguchi's robust parametric design involved crossing an orthogonal factorial design with control variables ($\mathbf{x}$) as its factors with another orthogonal factorial design with noise variables ($\mathbf{z}$) as its factors (Jones and Box, 1990). Shoemaker, Tsui, and Wu (1991) developed an alternative methodology that involves using both control variables ($\mathbf{x}$) and noise variables ($\mathbf{z}$) in the same factorial experiment. They proposed a combined array design that put both the control variable and noise variable in a single orthogonal factorial design. In a combined array design, fitting a model for the mean and variance with both $\mathbf{x}$ and $\mathbf{z}$ as the regressors has been called a response model approach (Myer et al., 1992; Borror et al., 2002). Both the mean and variance function can be described by a parametric model in this approach (Vining and Myer, 1990) Myer et al. (1997) indicated that the response model approach avoids unnecessary biases that appear in main effect estimates because certain interactions are ignored in the Taguchi's approach.

Vining and Bohn (1998) indicated that traditional parametric models can be inadequate for modeling the variance of response model when the parametric model cannot be adequately specified. They suggested using non-parametric techniques for estimating the variance of the model. Unfortunately, non-parametric techniques results in highly variable and biased estimates when data are sparse. To overcome this weakness, Pickle, Robinson, Birch and Anderson-Cook (2006) proposed a semi-parametric method in conjunction with the robust parameter design to

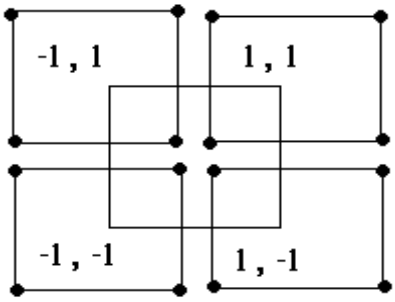
combine parametric and nonparametric estimation techniques to improve the estimates of both the mean and the variance of the responses.

The remaining chapters of the report are organized as follow. Chapter 2 describes Taguchi's robust parameter design for off-line quality control. In Chapter 3, we present the response model approach, which is the alternate method. Chapter 4 describes the semi-parametric technique.

CHAPTER 2 - Taguchi's Robust Parameter Design

In the 1980s, Taguchi introduced a robust parameter design. The robust parameter design consists of two types of factors: the control factors (**x**) that directly affect the quality and makeup of the product and the noise factors (**z**) that affect or change the quality of the products when they are used outside of the factory (Myers et al., 1992). Using both control and noise factors, Taguchi designed experiments to produce products with quality as close to the specifications as possible and as uniform as possible. The experimental design is called a crossed array design because it makes use of what Taguchi called the inner and outer arrays. The inner array contains control factors (**x**), while the outer array contains the noise factors (**z**). In this design, Taguchi crossed the inner array, an orthogonal design (e.g., 2^k factorial design) containing control factors (**x**), with the outer array, another orthogonal array that containing the noise factors (**z**)(Shoemaker et al., 1989; Box and John, 1990). If the inner array consists of n_1 runs and the outer array consists of n_2 runs, the design has a total of $n_1 \times n_2$ runs. For example, in a $2^2 \times 2^2$ crossed array design, there are each two factors with two levels in both the inner and outer array. This requires 16 runs for the experiment. Figure 2-1 depicts graphical example of such a crossed array design. In Figure 2-1, the corners of the inner array are symbolized by (-1,-1),

Figure 2-1 The $2^2 \times 2^2$ crossed array



(-1,1),(1,-1), and (1,1) for the control factors. Each outer array is a 2^2 factorial of the noise factors. (Myers et al., 1992; Box and John, 1990; Myer and Montgomery, 2002). Using crossed arrays, Taguchi's data collection plan is displayed in Figure 2-2. In Figure 2-2, $w_i = (w_{i1}, w_{i2}, ..., w_{il})$, $(i = 1,2,...., n_1)$ is the ith combination of the levels of the noise factors representing environmental conditions used in the field induced by the l noise factors. Similarly, $x_i = (x_{i1}, x_{i2}, ..., x_{ik})$, $(i = 1,2,...., n_2)$ is the ith combination for the levels of control factors. Let $Z(x_i), (i = 1,2,...., n_1)$ is the value of the performance statistics calculated over the n_2 noise conditions ($w_i, (i = 1,2,...., n_2)$) for the design point x_i of the inner array of the control factors. By using different performance statistics $Z(x_i)$, the analysis can provide insight as to which control variables affect the variance ($\sigma^2(x)$) and which affect the mean ($\eta(x)$). Ultimately, we will determine a setting of the control factors that yield mean to target with small variance. For example, using $Z_\sigma(x_j) = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(y_{ji} - \bar{y}_j)^2$ as a performance statistic, one can perform

Figure 2-2 Design Table

Control Factor	Noise Factor					Performance measure
	w_1	w_2	.....	.....	w_{n2}	
$x_1 = (x_{1\,1}, x_{1\,2}, ..., x_{1\,k})$	$y_{1\,1}$	$y_{1\,2}$	....	....	$y_{1\,n2}$	$Z(x_1)$
$x_2 = (x_{2\,1}, x_{2\,2}, ..., x_{2\,k})$	$y_{2\,1}$	$y_{2\,2}$	....	....	$y_{2\,n2}$	$Z(x_2)$
⋮ ⋮ ⋮	⋮		y_{ji}		⋮	⋮
⋮ ⋮ ⋮	⋮		⋮		⋮	⋮
$x_{n1} = (x_{n1\,1}, x_{n1\,2}, ..., x_{n1\,k})$	$y_{n1\,1}$	$y_{n1\,2}$	....	....	$y_{n1\,n2}$	$Z(x_{n1})$

an analysis to determine which setting of the control factors that will give the smallest variance of Y. Similarly, one can use $Z_\eta(x_j) = \frac{1}{n_2}\sum_{i=1}^{n_2} y_{ji}$ as a performance statistics to determine which setting of the control factors give the mean of Y that is closest to the target or the largest (or smallest). (Pignatiell, 1988)

In Taguchi's robust parameter design, he proposed finding an optimal setting of the levels of the control factors that minimized some loss function averaged over the noise factor space. A loss function reflects the loss that is incurred to society when the product's quality characteristic deviates from the target value. We can denote the loss function by $l(Y \mid x)$ and the expected loss by $E_{\Omega}[l(y \mid x)]$ where $E_{\Omega}(\cdot)$ represents the expectation taking over the noise factor space. In the case of the quadratic loss function: $l(y \mid x) = k[(Y \mid x) - \tau]^2$, where τ is a nominal target value and k is a proportionality constant, the expected loss is $E_{\Omega}[l(y|x)]=E_{\Omega}[k((Y|x)-\tau)^2]$. (Pignatiell, 1988; Shoemaker et al.,1991)

To minimize the expected loss $E_{\Omega}[l(y|x)]=E_{\Omega}[k((Y|x)-\tau)^2]$, Taguchi proposed several performance statistics that he called the signal-to-noise ratios which provide information about the expected loss. The three most commonly used signal-to-noise ratios of Taguchi are as follow; 1. If small response values are desired, Taguchi recommended the used of the signal-to-noise ratio: $SNR_s(x_j) = -10\log\sum_{i=1}^{n}\frac{y_{ji}^2}{n}$. 2. If large response values are desired, he recommended the signal-to-noise ratios: $SNR_L(x_j) = -10\log\sum_{i=1}^{n}\frac{1/y_{ji}^2}{n}$. 3. If it is desired to be close to a finite target value which is assumed to be zero, he recommended the use of signal-to-noise as: $SNR_T(x_j) = 10\log[\bar{y}^2(x)/s^2(x)]$, where $\bar{y}(x_j) = \sum_{i=1}^{n}\frac{y_{ji}}{n}$ and $s^2(x_j) = \sum_{i=1}^{n}\frac{[y_{ji} - \bar{y}(x_j)]^2}{n-1}$. (Pignatiell, 1988; Myers et al., 1992). In the use of signal-to-noise ratio in the typical analysis of promoted by Taguchi, one uses the marginal means analysis and the marginal means plots. We will use example 2.2(Myer et. al.,1992;Borror et. al., 1999;Robinson et. al.,2003) given below to illustrate this method of analysis.

Example 2.1. Taguchi's Robust Parameter Design (Box and John, 1990)

In this example, we considered an experiment for making a good tasting cake. The taste score on a scale of 1 through 7 is provided by five certified sensory specialists. Each score is the average score of the taste panel ratings of the quality of the cakes. The objective of this experiment is to find a recipe that can produce a reasonably good tasting cake following the baking instructions and recipe suggested by the manufacturer. The experiments are run with five factors: flour (F), shortening(S), egg (E), baking time (time), and baking temperature (Temp). The flour (F), shortening(S), and egg (E) are control factors. Time and temperature of baking are chosen as noise variables. It is difficult to bake a cake with the exact baking temperature and time as recommended by the instructions on the box because the temperature indicator and timer on a typical stove may not be accurate or people just don't follow the instruction carefully. Hence, this experiment design is a $2^3 \times 2^2$ cross array design consisting 2^3 factorial design for the control variables and 2^2 factorial design for the noise variables. Table 2-1 shows the levels of control factors and noise factors. The levels for the control variables are coded as -1 and 1 to represent low and high. The noise variable levels are coded similarly. Recipe '0' produces reasonably good tasting cake if the cake is baked at the exact baking temperature and time recommended by the manufacture (denoted by (0,0)). However, it makes poor tasting cake when baking temperature and time deviate from the recommended setting.

Table 2-1 Cake mix data

Recipe	Control Factors			Noise Factors					$\sum(y-7)^2$	$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
	F	S	E	Temp: 0	-	+	-	+		
				time : 0	-	-	+	+		
0	0	0	0	6.7	3.4	5.4	4.1	3.8	34.26	4.68
1	-	-	-	3.1	1.1	5.7	6.4	1.3	84.56	3.52
2	+	-	-	3.2	3.8	4.6	4.3	2.1	61.74	3.6
3	-	+	-	5.3	3.7	5.1	6.7	2.9	34.29	4.74
4	+	+	-	4.1	4.5	6.4	5.8	5.2	19.7	5.2
5	-	-	+	5.9	4.2	6.8	6.5	3.5	21.59	5.4
6	+	-	+	6.9	5.0	6.0	5.9	5.7	7.91	5.9
7	-	+	+	3.0	3.1	6.3	6.4	3.0	48.06	4.4
8	+	+	+	4.5	3.9	5.5	5.0	5.4	24.67	3.9

In the Table 2-1, we can see that recipe (6) not only has the highest average taste score ($\bar{y}$ =5.9) but also produces cakes scored consistently close to 7 even though the baking temperature and time are different from the recommended setting. Recipe (6) also has the lowest sum of squared error of taste score. This suggests that recipe (6) is the recipe of choice which is robust against affect of the noise factors.

Example 2.2 Wave soldering optimization

In an experiment described by Schmidt and Laundsby(1990), solder process optimization is performed by robust parameter design in a printed circuit board assembly plant. After the components are inserted into a bare board, the board is put through a wave solder machine which mechanically and electrically connects all the components into the circuit. Boards are placed in a conveyor and then put through the following steps: bathing in a flux mixture to remove oxide, preheating to minimize warpage, and soldering. An experiment is designed to determine the

conditions that give the minimum numbers of solder defects per million joints. The control factors and their levels are shown in the table below:

Control Factor	(-1)	(+1)
A, solder plot temperature(°F)	480	510
B, conveyor speed (ft/min)	7.2	10
C, flux density	0.9	1.0
D, preheat temperature(°F)	150	200
E, wave height(in.)	0.5	0.6

In this design, three noise factors are not easy to control in the process. They are the solder pot temperature, the small conveyor speed, and the assembly type. These noise factors can deviate from their nominal values inducing variability which can be transmitted to the response. Based on past experience, it is known that temperature varies within ± 5°F and that small conveyor speed varies within ±0.2 ft/min. Therefore, it is probable that variability can be increased greatly because the lack of capability to control these two factors at nominal levels. The third noise factor is the assembly type. The noise factors and their levels are shown in the table below;

Noise Factor	(-1)	(+1)
F, solder pot temperature (°F)	5	-5
G, small conveyor speed (ft/min)	+0.2	-0.2
H, assembly type	1	2

Both the control array (inner array) and the noise array (outer array) were chosen to be fractional factorials. The inner array is a 2^{5-2} design, and the outer array is a 2^{3-1}. The crossed array and the response values are shown Table 2-2.

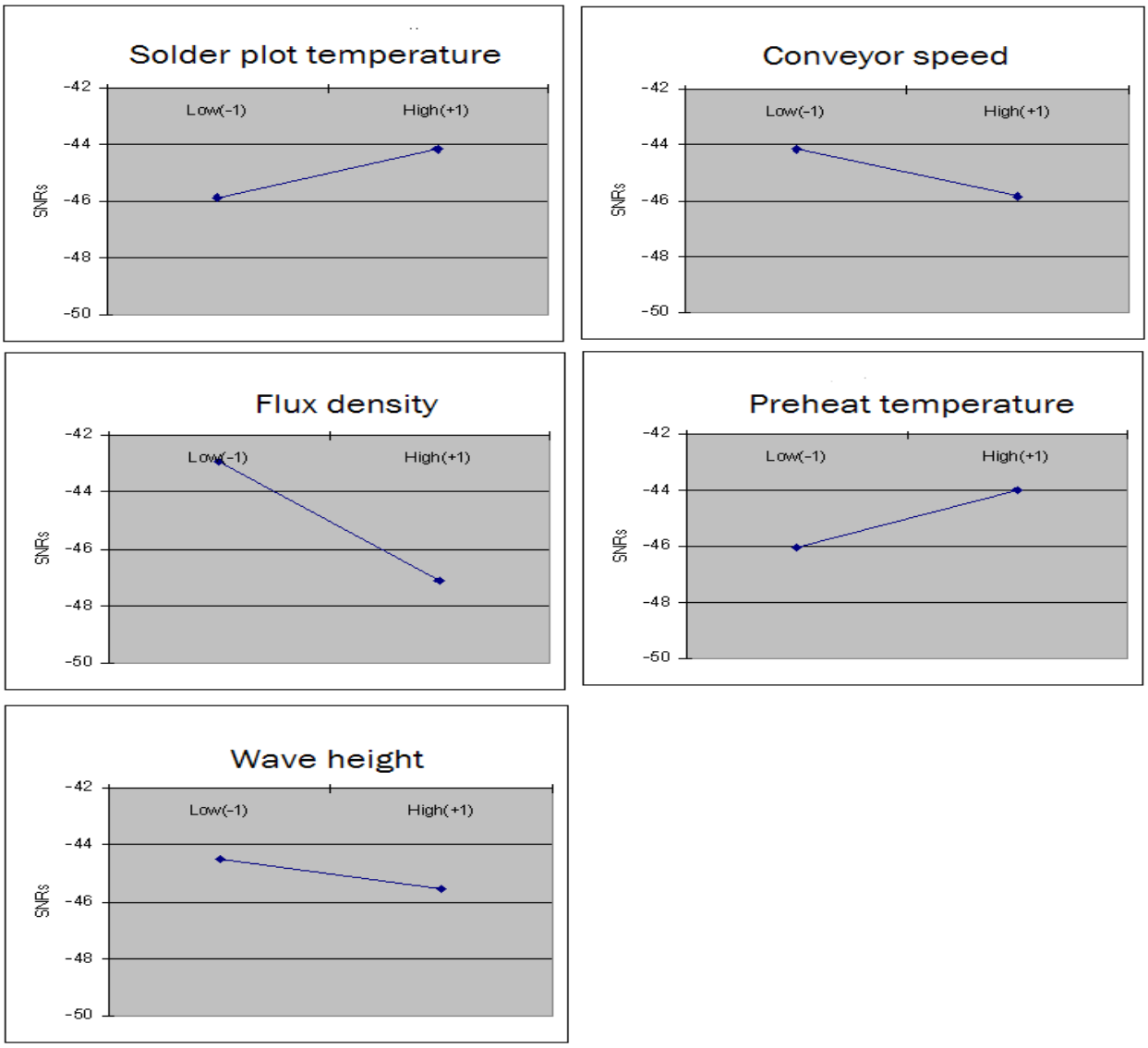
Table 2-2 The Wave Solder Experiment

Control Factors					Noise Factors						
					E	-1	1	1	-1		
					F	-1	1	-1	1		
A	B	C	D	E	G	-1	-1	1	1	$\bar{y}$	SNR_s
1	1	1	-1	-1		194	197	193	275	214.75	-46.75
1	1	-1	1	1		136	136	132	136	135.00	-42.61
1	-1	1	-1	1		185	261	264	264	243.5	-47.81
1	-1	-1	1	-1		47	125	127	42	85.25	-39.51
-1	1	1	1	-1		295	216	204	293	252	-48.15
-1	1	-1	-1	1		234	159	231	157	195.25	-45.97
-1	-1	1	1	1		328	326	247	322	305.75	-45.76
-1	-1	-1	-1	-1		186	187	105	104	145.5	-43.59

The Table 2-2 contains the means ($\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$) and the signal-to-noise ratios,

SNR_s ($SNR_s(x_j) = -10\log\sum_{i=1}^{n}\frac{y_{ji}^2}{n}$) computed at each design point of the inner array defined by the levels of the control factors. Since the experiment goal is to find the conditions that give minimum numbers of solder defects, researchers want the mean ($\bar{y}$) to be small and the signal-to-noise ratio to be large. For instance, the first row SNR_s is -46.75 that is calculated to be $-10\log\left(\frac{(194)^2+(197)^2+(193)^2+(275)^2}{4}\right)$. In the Figure 2-3, the means of SNR_s is plotted against the levels of each control factor. The means are taken across levels of the other factors. For example, at temperature, 510 °F temperature (coded as +1) SNR_s averaged over the levels of other factors is -44.17($[(-46.75)+(-42.61)+(-47.81)+(-39.51)]/4$).Similarly, at temperature

Figure 2-3 Plot of SNRs against level for each control factor

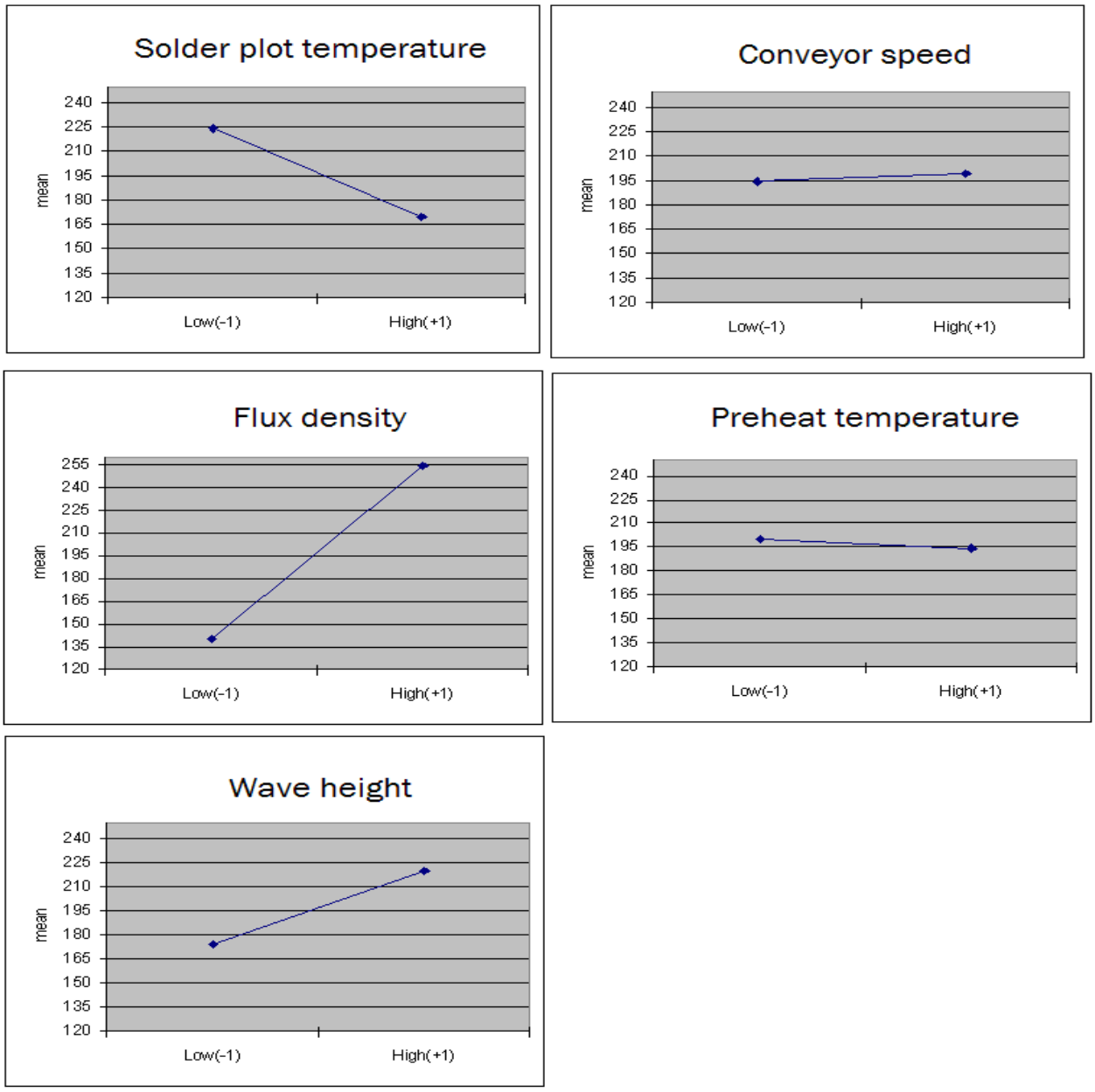


480 °F (coded as -1) SNR_s averaged over the levels of other factors is -45.86

$([(-48.15)+(-45.97)+(-45.76)+(-43.59)]/4)$. The average SNR_s of other control factors such as designing speed, preheat and wave height at high and low level are calculated exactly the same way. In the Figure 2-3, solder plot temperature and preheat temperature plots show increase from low(-1) to high(+1). However, the conveyor speed, flux density and wave height show decrease from low(-1) to high(+1), The flux density plot decreases from low to high dramatically.

In Figure 2-4, the marginal means ($\bar{y}$) of control factors are plotted against their respective levels. These means at high and low levels are computed in the same way as above. In the Figure 2-4, the conveyor speed, flux density and wave height plot show increase from low (-1) to high(+1). The conveyor speed increases slightly, but flux density increase dramatically. The solder plot temperature and preheat temperature plot illustrate decrease from low to high. We can see that solder plot temperature plot decrease dramatically.

Figure 2-4 A plot of $\bar{y}$ against the levels of control factors



From Figure 2-3 and 2-4, we can see that temperature and flux density are critical control factors because each of their plots illustrate considerably change from low(-1) to high(+1). It also appears that wave height has some influence on SNR_s but little more on $\bar{y}$. In Taguchi's signal-to-noise ratio, it is of interest to maximize signal-to-noise to diminish the expected loss. Hence, in this experiment, we are interested in minimizing $\bar{y}$ and maximizing SNR_s. The suggested operating conditions are

Solder plot temperature = 510°F
Flux density = 0.9
Wave height = 0.5 in.

The conveyor speed and preheat temperature can be placed at the most economical settings, presumably at low levels. The effect of wave height is marginal compared to the impact of solder plot temperature and flux density. The analysis suggests that the conditions given above are the most robust conditions, that is, in the context of this example, those conditions that are most insensitive to changes in the noise factors.

Although Taguchi developed the method of robust parameter design, many authors(Shoemaker, Tsui, and Wu,1991; Nair et al., 1992; Borror and Montgomery, 1999) have indicated that his approach has drawbacks. First, Shoemaker, Tsui, and Wu (1991) noted that his method required a large number of runs. They suggested that this method is not cost-effective because the noise factor array is repeated for every row in the control factor array. Second, Nair et al. (1992) indicated that the Taguchi approach focused on the signal-to-noise ratio as a performance measure. Taguchi's approach to data analysis begins by defining a signal-to-noise

ratio and then seeks a model for it in terms of the experimental factors. Nair et al. mentioned that this method could lead to a great loss of information in the statistical sense and fail to use all of the information in the data. Third, Shoemaker et al. (1991) and Borror et al. (1999) showed that Taguchi's method could, only with difficulty, take into account the interaction. They mentioned that there is no flexibility to estimate control factor interactions because of the crossed array structure.

CHAPTER 3 - Response Model Approach

Shoemakere, Tsui, and Wu (1991) developed an alternative method that overcome the weaknesses of Taguchi method. They suggested a combined array design where both control variables ($\mathbf{x}$) and noise variables ($\mathbf{z}$) are arranged in the same experiment. They also introduced the response model approach that uses both control variables ($\mathbf{x}$) and noise variables ($\mathbf{z}$) in the same model (Myer et al., 1992; Borror et al., 2002). A combined array design requires fewer runs than Taguchi's crossed array design and more readily allows estimation of potentially important interactions between the control variables. The response model approach avoids unnecessary biases that appear in main effect estimates due to ignoring interactions of the control variables. These biases can be substantial when the experimental design is highly fractionated (Myer et al., 1997). The response model can take many forms, but we discuss here the model described by Box and Jones (1990) and further elaborated by Myers, Khuri, and Vining (1992):

$$y(\mathbf{x},\mathbf{z}) = \beta_0 + \mathbf{x}'\boldsymbol{\beta} + \mathbf{x}'\mathbf{B}\mathbf{x} + \mathbf{z}'\boldsymbol{\gamma} + \mathbf{x}'\boldsymbol{\Delta}\mathbf{z} + \varepsilon .$$

In this model, we assume r_1 controllable variables $\mathbf{x}' = [x_1, x_2 \cdots\cdots x_{r_1}]$ and r_2 noise variables $\mathbf{z}' = [z_1, z_2 \cdots\cdots z_{r_2}]$. This model contain a full quadratic model for the control variables $(\beta_0 + \mathbf{x}'\boldsymbol{\beta} + \mathbf{x}'\mathbf{B}\mathbf{x})$, the main effect terms for the noise variables $(\mathbf{z}'\boldsymbol{\gamma})$, and all control-factor by noise-factor interactions $(\mathbf{x}'\boldsymbol{\Delta}\mathbf{z})$. $\boldsymbol{\beta}$ is an $r_1 \times 1$ vector of the regression coefficients of control factors; $\mathbf{B}$ is an $r_1 \times r_1$ matrix, whose main diagonals are regression coefficients for the squared terms of the control factors, and the off-diagonals are one-half of regression coefficients for the interaction of the control factors; $\boldsymbol{\gamma}$ is an $r_2 \times 1$ vector of the regression coefficients for the main effects of the noise variables; and $\boldsymbol{\Delta}$ is a $r_1 \times r_2$ matrix of regression coefficients for the

control-by-noise interaction effects. In this model, ε is often assumed to be NID$(0, \sigma^2)$ and noise factors have been scaled so that they have $E(\mathbf{z}) = \mathbf{0}$, $Var(\mathbf{z}) = \mathbf{I}_r$. The model proposed by Myers, Khuri, and Vining (1992) can be considered as a model for the mean response conditioned on $\mathbf{z}$. The model for the unconditional mean is

$$E\ [y(\mathbf{x},\mathbf{z})] = \beta_0 + \mathbf{x}'\boldsymbol{\beta} + \mathbf{x}'\mathbf{B}\mathbf{x}\ .$$

Similarly, a model for the response variance is

$$Var_z[y(\mathbf{x},\mathbf{z})] = Var_z[(\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta})\mathbf{z}] + \sigma^2\ .$$

In this equation, the quantity, $\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta} = \mathbf{a}'$, is a vector of constants. Since $Var_z(\mathbf{a}'\mathbf{z}) = \mathbf{a}'Var(\mathbf{z})\mathbf{a}$ and $Var(\mathbf{z}) = \mathbf{I}$, we have

$$Var_z(\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta})\mathbf{z} = (\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta})(\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta})'\ .$$

Therefore,

$$Var_z[y(\mathbf{x},\mathbf{z})] = (\boldsymbol{\gamma}' + \mathbf{x}'\boldsymbol{\Delta})(\boldsymbol{\gamma} + \mathbf{x}\boldsymbol{\Delta}') + \sigma^2\ .$$

We note that $\boldsymbol{\gamma} + \mathbf{x}\boldsymbol{\Delta}'$ is the vector of partial derivatives of $y(\mathbf{x},\mathbf{z})$ with respect to the noise factors **z**. Thus, $\boldsymbol{\gamma} + \mathbf{x}\boldsymbol{\Delta}'$ is the slope of the response surface in the direction of the noise factors.

In practice, the experimenter fits the response model using data from 2^k factorial experimental design. This yields an estimation of $E[y(\mathrm{x},\mathrm{z}) \mid z]$, the conditional expectation of Y given Z . The response model conditioned on $\mathbf{z}$ would be fitted to the data using least squares. This results in a fitted model

$$\hat{y}(\mathbf{x},\mathbf{z}) = \hat{\beta}_0 + \mathbf{x}'\hat{\boldsymbol{\beta}} + \mathbf{x}'\hat{\mathbf{B}}\mathbf{x} + \mathbf{z}'\hat{\mathbf{r}} + \mathbf{x}'\hat{\boldsymbol{\Delta}}\mathbf{z}\ .$$

Consequently, the estimated process mean and variance are given respectively by

$$\hat{E}\ [y(\mathbf{x},\mathbf{z})] = \hat{\mu}_z[y(\mathbf{x},\mathbf{z})] = \hat{\beta}_0 + \mathbf{x}'\hat{\boldsymbol{\beta}} + \mathbf{x}'\hat{\mathbf{B}}\mathbf{x}\text{, and}$$

$$\hat{Var}_z[y(\mathbf{x},\mathbf{z})] = (\hat{\boldsymbol{\gamma}}' + \mathbf{x}'\hat{\boldsymbol{\Delta}})(\hat{\boldsymbol{\gamma}} + \mathbf{x}\hat{\boldsymbol{\Delta}}') + \hat{\sigma}^2.$$

Here, $\hat{\sigma}^2$ is the mean square error from the fitted response model. Using these equations the standard robust parameter design (RPD) problems can be formulated. One can apply an appropriate constrained optimization procedure to obtain recommended settings that maximize (or minimize) $\hat{E}\ [y(\mathbf{x},\mathbf{z})]$ and minimize $\hat{Var}\ [y(\mathbf{x},\mathbf{z})]$ (Borror et al., 2002;Myer et al., 1997;Myer et al., 1992; Borror and Montgomery, 1999). The following example 3.1 shows how to estimate the process mean and variance and how to determine the optimal setting of a process.

Example 3.1. The pilot plant experiment.

Montgomery (2001) described a factorial experiment carried out in a pilot plant to study factors thought to influence the filtration rate of a chemical product. The experimental objective was to find factor settings to maximize the filtration rate while keeping the variation of the process as low as possible. The four factors are temperature, pressure, concentration of formaldehyde, and stirring rate. Each factor is present at two levels (-, +), and the data obtained from a single replicate of the 2^4 experiment are shown in Table 3-1. Because temperature is hard to control in the experiment, it is designated as the noise variable and denoted by (z_1). The other three factors are control factors: pressure (x_1), concentration of formaldehyde (x_2), and stirring rate (x_3). Because both the control factors and the noise factor are in the same 2^4 factorial design, this type of design is called a combined array design.

Table 3-1 Pilot plant filtration rate experiment

Run number	Factor				Filtration rate(gal/hr)
	z_1	x_1	x_2	x_3	
1	-	-	-	-	45
2	+	-	-	-	71
3	-	+	-	-	48
4	+	+	-	-	65
5	-	-	+	-	68
6	+	-	+	-	60
7	-	+	+	-	80
8	+	+	+	-	65
9	-	-	-	+	43
10	+	-	-	+	100
11	-	+	-	+	45
12	+	+	-	+	104
13	-	-	+	+	75
14	+	-	+	+	86
15	-	+	+	+	70
16	+	+	+	+	96

The PROC RSREG was used to estimate the response model conditioned on **z** and the results are shown in Table 3-2.

Table 3-2 PROC RSREG results.

Term	F	Coeff.	STD Error	t Value	Pr > |t|
Intercept	**1**	**70.062500**	**1.263984**	**55.43**	**<.0001**
$\mathbf{z}_1$	**1**	**10.812500**	**1.263984**	**8.55**	**0.0004**
x_1	1	1.562500	1.263984	1.24	0.2713
$\mathbf{x}_2$	**1**	**4.937500**	**1.263984**	**3.91**	**0.0113**
$\mathbf{x}_3$	**1**	**7.312500**	**1.263984**	**5.79**	**0.0022**
$x_1 \times z_1$	1	0.062500	1.263984	0.05	0.9625
$\mathbf{x}_2 \times \mathbf{z}_1$	**1**	**-9.062500**	**1.263984**	**-7.17**	**0.0008**
$x_2 \times x_1$	1	1.187500	1.263984	0.94	0.3906
$\mathbf{x}_3 \times \mathbf{z}_1$	**1**	**8.312500**	**1.263984**	**6.58**	**0.0012**
$x_3 \times x_1$	1	-0.187500	1.263984	-0.15	0.8879
$x_3 \times x_2$	1	-0.562500	1.263984	-0.45	0.6749

The Table3-2 shows that the main factors(x_2, x_3 and z_1) are significant and interactions($x_2 \times z_1$ and $x_3 \times z_1$) are also significant at a significance level of 0.05. A final model can be developed using only the significant factors and interactions:

$$\hat{y}(\mathbf{x}, z_1) = 70.06 + 10.81z_1 + 4.94x_2 + 7.31x_3 - 9.06x_2z_1 + 8.31x_3z_1 .$$

Suppose that the above fitted model allows us to assume that the true relationship between y, **x** and z_1 is $\hat{y}(\mathbf{x}, z_1) = \beta_0 + \delta_1 z_1 + \beta_2 x_2 + \beta_3 x_3 - \delta_{12} z_1 x_2 + \delta_{13} z_1 x_3 + \varepsilon$.

Because $E(z_1) = 0$ and $E(\varepsilon) = 0$, a model for the process mean is

$$E[y(\mathbf{x}, z_1)] = \beta_0 + \beta_2 x_2 + \beta_3 x_3 .$$

The process variance is

$$Var[y(\mathbf{x}, z_1)] = Var_{z_1}[\beta_0 + \gamma_1 z_1 + \beta_2 x_2 + \beta_3 x_3 + \delta_{12} z_1 x_2 + \delta_{13} z_1 x_3 + \varepsilon]$$
$$= Var_{z_1}[\beta_0 + \beta_2 x_2 + \beta_3 x_3 + (\gamma_1 + \delta_{12} x_2 + \delta_{13} x_3) z_1 + \varepsilon] = (\gamma_1 + \delta_{12} x_2 + \delta_{13} x_3)^2 + \sigma^2.$$

We can replace the parameters by their estimates to obtain

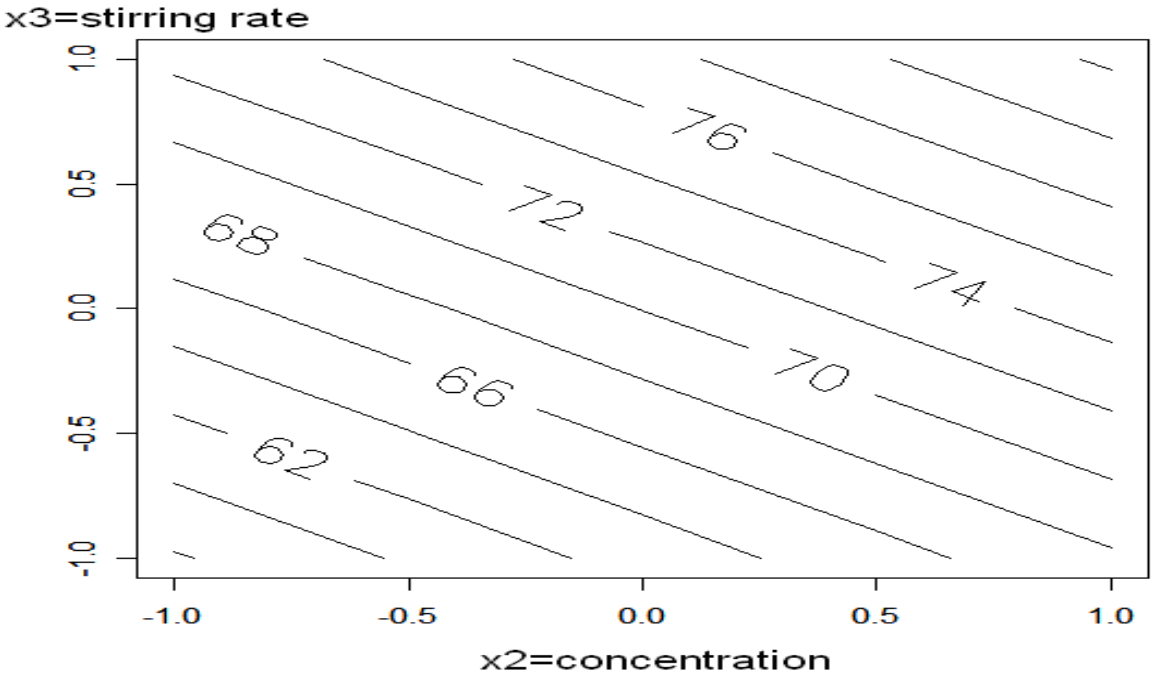
$$\hat{E}\ [y(\mathbf{x}, z)] = 70.06 + 4.94 x_2 + 7.31 x_3,$$

The SAS output provides σ^2 =19.51. The estimated variance function is

$$\hat{Var}[y(\mathbf{x}, z)] = \sigma_z^2 (10.81 + 9.06 x_2 + 8.31 x_3)^2 + \sigma^2$$
$$= 1 * (10.81 + 9.06 x_2 + 8.31 x_3)^2 + 19.51$$
$$= 136.42 + 82.08 x_2^2 + 69.06 x_3^2 - 195.88 x_2 + 179.66 x_3 - 150.58 x_2 x_3.$$

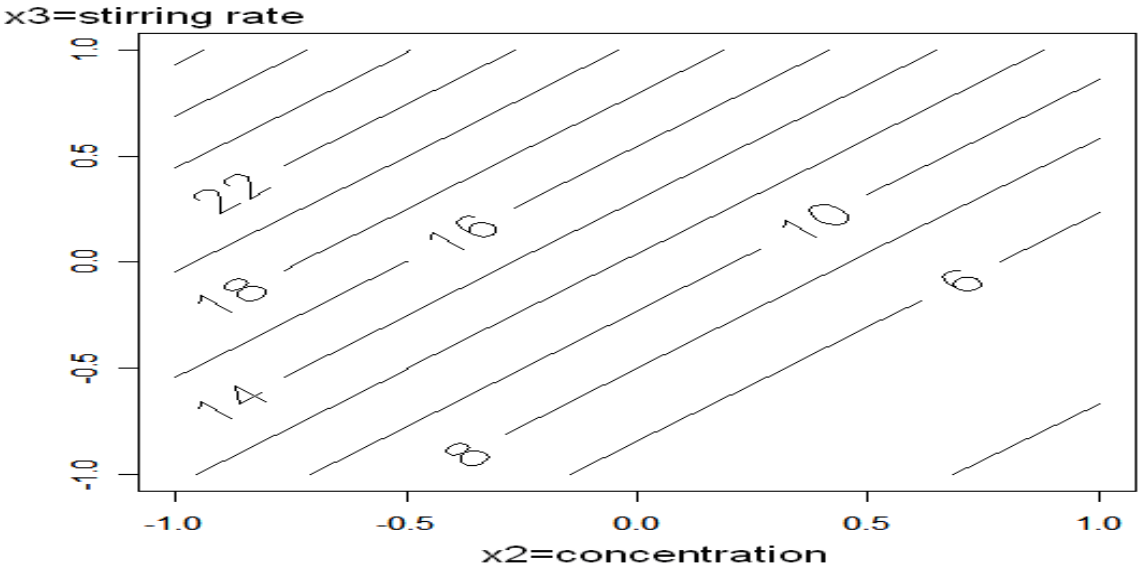
Figure 3-1 presents a contour plot of the response from the mean model ($\hat{E}\ [y(\mathbf{x}, z)] = 70.06 + 4.94 x_2 + 7.31 x_3$). We can see that the mean filtration rate increases as either the concentration or the stirring rate increases.

Figure 3-1 Contour of mean filtration rate

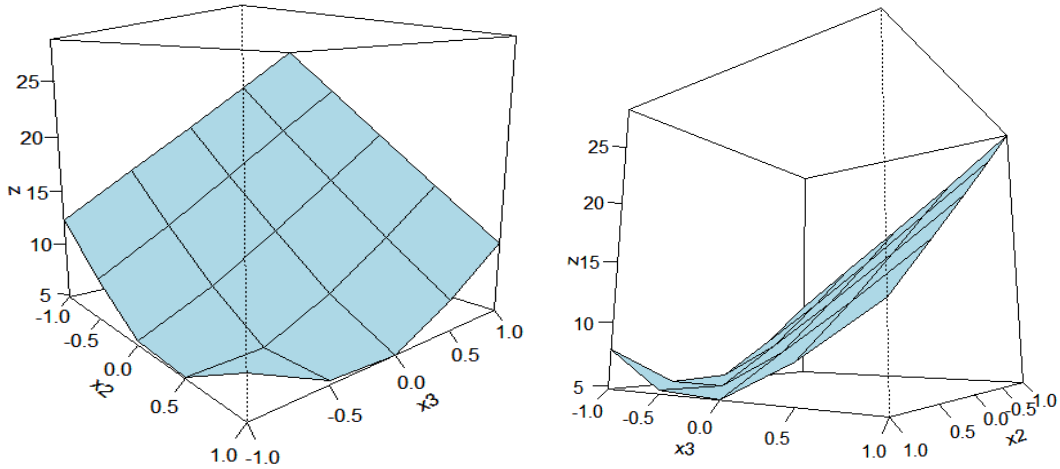


We also constructed plots of the square root of the variance contours, labeled propagation of error, or POE. POE is the standard deviation of the transmitted variability in the response as a function of the controllable variables. Figure 3-2 shows a contour plot and a three-dimensional response surface plot of the POE. In Figure 3-2, we can see that the POE decreases as concentration increases and stirring rate decreases.

Figure 3-2 Contour plot of propagation of error



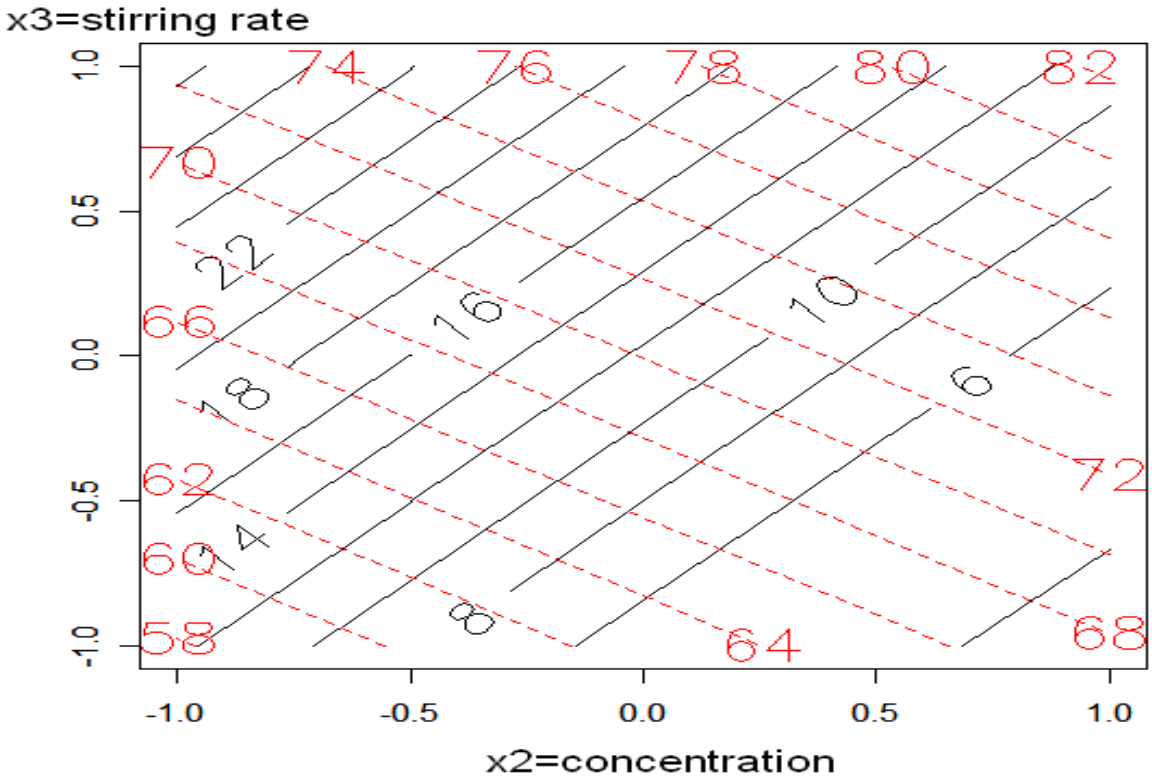
(a) Contour plot



(b) Response surface plot

Suppose that the researcher wants to maintain a mean filtration rate of about 72 and minimize the variability around this value. Figure 3-3 shows an overlay plot of the contours of mean filtration rate and that of the POE as a function of concentration and stirring rate, the significant control factors. It is clear from the plot in Figure 3-3 that it will be necessary to hold concentration at the high level and stirring rate very near the middle level to achieve the desired objectives.

Figure 3-3 Overlay plot of the contours of the mean filtration rate and the POE



Vining and Bohn (1998) indicated a shortcoming of this approach because it assumes a parametric model for the response which provides a way to estimate the process variance and process mean. In practice, people often have difficulty modeling the process variance with a parametric model because the process variance typically is rather noisy. Therefore, this method and any parametric method of estimating the process variance may not provide a good estimate of the process variance.

CHAPTER 4 - Semi-Parametric Approach Method

4.1. Introduction

In parametric[1] and nonparametric[2] approaches for fitting the process mean and variance, the parametric method provides superior fit if the underlying functions can be effectively expressed parametrically and if the researcher properly specifies the parametric forms. The parametric estimates, however, may be highly biased, and the optimal control factor settings can be miscalculated if the models are not correctly specified. In contrast, if researchers don't have any information about the form of underlying functions, the nonparametric method provides a very useful alternative. Nonparametric methods can offer superior fit by capturing structure in the data that a misspecified parametric model cannot. However, nonparametric methods were initially developed for situations with large sample sizes. Hence nonparametric fitting has some problems with small sample sizes. For small sample sizes, nonparametric fitting may be unstable because estimated mean and variance functions are highly variable. Many authors (Einsporn and Born, 1993; Mays et al., 2001; Robinson et al., 2002; Pickle et al., 2006) introduced the semi-parametric approaches that combine the parametric and nonparametric methods. This method offers estimated functions that have less bias than parametric approaches and less variance than nonparametric approaches in the cases they considered. Hence, the semi-parametric approach offers a viable alternative to the parametric approach when the mean and variation function cannot be adequately specified.

[1] See 4.2.Parametric Approach

[2] See 4.3. Nonparametric Approach

4.2. Parametric Approach

A model for the process mean linear in the model parameters can be written as $\bar{y}_i$ [3] $= h(x_i) + g^{1/2}(z_i\gamma)\varepsilon_i = \mathbf{x}_i'\boldsymbol{\beta} + g^{1/2}(\mathbf{x}_i^{*'};\boldsymbol{\gamma})\varepsilon_i$, where $x_i^{'}$ and $x_i^{*'}$ are $1\times k$ and $1\times l$ vectors of mean and variance model regressors, respectively, is expanded to linear model form, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are $k\times 1$ and $l\times 1$ vectors of mean and variance model parameters, respectively, g is the underlying variance function, and ε_i denotes the random error for the model. The ε_i's are assumed to be uncorrelated with mean zero and variance of one.

In variance modeling the log-linear model proposed by Bartlett and Kendall (1946) is a popular one, written explicitly as $\ln(s_i^2)$ [4] $= g^*(\mathbf{x}_i^*) + \eta_i = \mathbf{x}_i^{*'}\boldsymbol{\gamma} + \eta_i$ where the η_i's are independent model error terms whose expectation is assumed to be zero and whose variance is assumed to be constant across the d [5] design points.

Assuming the model forms for the mean and variance given above, the model parameters are estimated using the following estimated weighted least squares (EWLS) algorithm:

Step1 : Fit the variance model, $\ln(s_i^2) = \mathbf{x}_i^{*'}\boldsymbol{\gamma} + \boldsymbol{\eta}_i$, via ordinary least square(OLS), obtaining $\hat{\gamma}^{(OLS)} = (\mathbf{X}^{*'}\mathbf{X}^*)^{-1}\mathbf{X}^*\mathbf{y}^*$ where $\mathbf{y}^*$ is the $d\times 1$ vector of log transformed sample variances and $\mathbf{X}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, ..., \mathbf{x}_d^*)^{/}$. Denote $\hat{\mathbf{y}}^{*(OLS)} = \mathbf{X}^*\hat{\boldsymbol{\gamma}}^{(OLS)}$ and $\hat{\mathbf{y}}_i^{*(OLS)} = \mathbf{x}_i^{*'}\hat{\boldsymbol{\gamma}}^{(OLS)}$.

---

[3] See page 5

[4] See page 5

[5] See page 5

Step 2: Use $\hat{\sigma}_i^2 = \exp(\mathbf{x}_i^{*'}\hat{\boldsymbol{\gamma}}^{(OLS)}) = \exp(\hat{\mathbf{y}}_i^{*})^{(OLS)}$ as the estimated variances to compute the $d \times d$ estimated variance-covariance matrix for the means model, $\hat{\mathbf{V}} = diag(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \ldots, \hat{\sigma}_d^2)$ if $n_i = n$ for $i = 1,2,.....,d$.

Step 3: Use $\hat{\mathbf{V}}^{-1}$ as the estimated weight matrix to fit the means model, yielding $\hat{\boldsymbol{\beta}}^{(EWLS)} = (\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{V}}^{-1}\overline{\mathbf{y}}$, where $\overline{\mathbf{y}}$ denotes the $d \times 1$ vector of sample averages and $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_d)^{/}$

The algorithm above yields the following estimates of the process mean and variance functions:

Estimated process mean: $\hat{E}[y_i]^{(EWLS)} = \mathbf{x}_i^{'}\hat{\boldsymbol{\beta}}^{(EWLS)}$

Estimated process variance: $\hat{Var}[y_i]^{(OLS)} = \exp(\mathbf{x}_i^{*'}\hat{\boldsymbol{\gamma}}^{(OLS)})$

If dispersion factors are present and these factors also influence the process mean, the researcher is left with finding the levels of the control factors that yield a desirable trade off between low variance and a small deviation from the targeted mean. This is often done via minimization of an objective function such as the square error loss (SEL):

$\text{SEL} = E[y(\mathbf{x}) - T]^2 = \{E[y(\mathbf{x})] - T\}^2 + Var[y(\mathbf{x})]$, where T denotes the target value for the process mean.

4.3. Nonparametric Approach

Here, h and g^* the mean and variance function are assumed to be unknown. Anderson-Cook and Prewitt (2005) used local polynomial regression in the nonparametric approach. The local polynomial regression (LPR) is a popular class of nonparametric smoothing methods and is

particular appealing in response surface applications due to its robustness to biased estimates at the boundary of the design space. LPR is essentially a weighted least squares (WLS) problem where the weights are given by a kernel function. Let the kernel function for estimating the mean function at point $\mathbf{x}_0 = (x_{01}, x_{02}, ...x_{0k})$ be of the form

$$\mathbf{K}(\mathbf{x}_0, \mathbf{x}_i) = \frac{1}{b^k} \prod_{j=1}^{k} K\left(\frac{x_{0j} - x_{ij}}{b} \right), \text{ where } \mathbf{x}_i = (x_{i1}, x_{i2}, ...x_{ik}),\ K(z), \text{(e.g. } K(z) = e^{-z^2}\text{) is a}$$

univariate kernel function, and b is the bandwidth. A different kernel function may be used for estimating the variance function as the regressors affecting the mean do not necessarily affect the variance.

Mays et al. (2001) introduced a penalized cross-validation technique, PRESS**, for choosing an appropriate bandwidth. The approach chooses the bandwidth as the value b that minimizes PRESS**, defined as:

$$PRESS** = \frac{PRESS}{d - trace(\mathbf{H}^{(LLR)}) + (d - (k+1)) \dfrac{SSE_{\max} - SSE_b}{SSE_{\max}}},$$

where $SSE_{\max}$ is the largest error sum of squares over all possible bandwidth values, SSE_b is the error sum of square associated with a particular bandwidth value b, k is the number of regressors, and the prediction error sum of squares, PRESS, is given by

PRESS = $\sum_{i=1}^{d} (y_i - \hat{y}_{i,-1})^2$, where $\hat{y}_{i,-1}$ denotes the estimated response obtained by leaving out the i^{th} observation when estimating at location $\mathbf{x}_i$. The LLR smoother matrix, $\mathbf{H}^{(LLR)}$, is

$\mathbf{H}^{(LLR)}=\begin{bmatrix}\mathbf{h}_1^{(LLR)'}\\ \mathbf{h}_2^{(LLR)'}\\ \vdots\\ \mathbf{h}_d^{(LLR)'}\end{bmatrix}$ with $\mathbf{h}_i^{(LLR)'}=x_i^{*'}(\mathbf{X}^{*'}\mathbf{Q}_i^{*}\mathbf{X}^{*})^{-1}\mathbf{X}^{*'}\mathbf{Q}_i^{*}\mathrm{y}^{*}$ where $\mathbf{Q}_i^{*}$ is the diagonal matrix of the kernel weight associated with $\mathbf{x}_i^{*}$ defined below.

Let $\mathbf{Q}_0=diag(\sqrt{q_{01,}}\sqrt{q_{02}},.....\sqrt{q_{0d}})$ where the $q_{0i}=\dfrac{\mathbf{K}(x_0,x_i)}{\sum_{i=1}^{d}\mathbf{K}(x_0,x_i)},(i=1,2,...,d)$ are the kernel weights associated with x_0. Similarly define $\mathbf{Q}_0^{*}=diag(\sqrt{q_{01}^{*}},\sqrt{q_{02}^{*}},.....\sqrt{q_{0d}^{*}})$ where the q_{0i}^{*}'s are the kernel weights associated with x_0^{*} in the estimation of the variance model. Then the LLR estimation of the mean and variance functions at $x=x_0$ are respectively

$\hat{y}_0^{(LLR)}=x_0^{'}(\mathbf{X}^{'}\mathbf{Q}_0\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{Q}_0y$, $\hat{\sigma}_0^2=\exp[x_0^{*'}(\mathbf{X}^{*'}\mathbf{Q}_0^{*}\mathbf{X}^{*})^{-1}\mathbf{X}^{*'}\mathbf{Q}_0^{*}y^{*}]=\exp[y_0^{*'(LLR)}]$.

Let $\mathbf{W}_0=\mathbf{Q}_0\hat{\mathbf{V}}^{-1}\mathbf{Q}_0$, where $\hat{\mathbf{V}}=diag(\hat{\sigma}_1^2,\hat{\sigma}_2^2,.....,\hat{\sigma}_d^2)$. Then the estimated weighted local linear regression(EWLLR) estimate of y_0 associated with $x=x_0$ is

$\hat{y}_0^{(EWLLR)}=x_0^{'}(\mathbf{X}^{'}\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{W}_0y$.

4.4. Semi-Parametric Approach

Einsporn and Birth (1993) proposed a semi-parametric method for modeling the mean response assuming constant error variance. They introduced model robust regression 1 (MRR1), which combines the ordinary least squares (OLS) fit and the local linear regression (LLS) fit to the raw data in a convex combination via a mixing parameter λ. For example, if $\hat{\mathbf{y}}^{(OLS)}$ denotes the vector of ordinary least square (OLS)[6] estimates of the mean and if $\hat{\mathbf{y}}^{(LLS)}$ denotes the local

[6] See 4.2.Parametric Approach.

linear regression (LLR)[7] estimates of the mean, then the model robust regression 1 (MRR1) estimate of the mean responses are obtained as $\hat{\mathbf{y}}^{(MRR1)} = \lambda\hat{\mathbf{y}}^{(LLR)} + (1-\lambda)\hat{\mathbf{y}}^{(OLS)}$, where $\lambda \in [0,1]$. Similar to the choice of bandwidth in local linear regression (LLR), the choice of mixing parameter, λ, involves a bias-variance trade-off. Mays et al. (2001) obtained the following expression for the asymptotically optimal value of the mixing parameter, λ, for MRR1:

$$\hat{\lambda}^{(MRR1)} = \frac{\left\langle \hat{\mathbf{y}}_{-1}^{(LLR)} - \hat{\mathbf{y}}_{-1}^{(OLS)}, \mathbf{y} - \hat{\mathbf{y}}^{(OLS)} \right\rangle}{\left\| \hat{\mathbf{y}}^{(LLR)} - \hat{\mathbf{y}}^{(OLS)} \right\|},$$

where the i^{th} observation of $\hat{\mathbf{y}}_{-1}^{(LLR)}$ and $\hat{\mathbf{y}}_{-1}^{(OLR)}$ are $\hat{\mathrm{y}}_{i,-1}^{(LLR)}$ and $\hat{\mathrm{y}}_{i,-1}^{(OLR)}$, respectively. The values $\hat{\mathrm{y}}_{i,-1}^{(LLR)}$ and $\hat{\mathrm{y}}_{i,-1}^{(OLR)}$ denote the local linear regression (LLR) and ordinary least square (OLS) estimates of y at $x = x_i$ obtained by leaving out the i^{th} observation. The notation $\langle\ \rangle$ represents the inner product and $\|\ \|$ represents the standard L_2 (Euclidean) norm. MRR1 provides a smooth estimate that captures important anomalies in the data, which parametric methods may not be able to model. However, if there are design points is in the data where both the parametric and nonparametric estimates are too high or too low, then MRR1 estimates will also be too high or too low because MRR1 estimates with a convex combination of $\hat{\mathrm{y}}^{(LLR)}$ and $\hat{\mathrm{y}}^{(OLR)}$. The MRR1 estimates of the log of the variance function can be obtained similarly.

Mays et al. (2001) introduced model robust regression 2 (MRR2) that overcomes this shortcoming. Their approach, like MRR1, combines a parametric fit and nonparametric fit via a mixing parameter. However for MRR2, the nonparametric fit is applied to the residuals obtained from the OLS parametric fit of the mean function. The vector of residuals (r) represents the structure in the data that is not captured by the user specified parametric model. The vector of

[7] See 4.3. Nonparametric Approach.

residuals is fit nonparametrically via LLR resulting in a vector of smoothed residuals $\hat{\mathbf{r}}$. MRR2 estimates are then obtained by adding a portion of the LLR smoothed residuals back to the original parametric fit, yielding: $\hat{\mathbf{y}}^{(MRR2)} = \hat{\mathbf{y}}^{(OLS)} + \lambda\hat{\mathbf{r}}$, where $\lambda \in [0,1]$. Mays et al. (2001) derived an expression for the asymptotically optimal mixing parameter. The expression is

$$\hat{\lambda}_{opt}^{(MRR2)} = \frac{\langle \hat{\mathbf{r}}, \mathbf{r} \rangle}{\|\hat{\mathbf{r}}\|^2}.$$

Notice that $\hat{\lambda}_{opt}^{(MRR2)} * \hat{\mathbf{r}}$ is essentially the projection of $\mathbf{r}$ on $\hat{\mathbf{r}}$. The MRR2 estimates of the log of the variance function can also be obtained similarly.

Mays et al. (2001) demonstrated that MRR2 provides better estimates for mean and variance models than MRR1. Robinson and Birch (2002) extend the MRR2 to the Dual Model Robust Regression (DMRR), which uses MRR1 to estimate variance and MRR2 to estimate the means model. They used the following algorithm to find the Dual Model Robust Regression.

Step 1: Fit the variance model via MRR1 to get the Variance Model Robust Regression (VMRR) estimate : $\hat{Var}[y]^{(VMRR)} = \exp[\lambda_\sigma \hat{\mathbf{y}}^{*(LLR)} + (1-\lambda_\sigma)\hat{\mathbf{y}}^{*(OLS)}]$,

where $\lambda_\sigma \in [0,1]$ is the asymptotically optimal variance model mixing parameter.

Step 2: Use the i^{th} component of $\hat{Var}(y^{VMRR}), \hat{\sigma}_i^2$ as the estimated variance of y_i. Then the variance-covariance matrix for the means model is estimated by $\hat{\mathbf{V}} = diag(\hat{\sigma}_{1,}^2 \hat{\sigma}_{2,}^2 .... \hat{\sigma}_d^2)$

Step 3: Use $\hat{\mathbf{V}}^{-1}$ as the estimated weight matrix to obtain the parametric estimate of the means model via estimated weighted least square (EWLS)[8]. EWLS yields

$\hat{\boldsymbol{\beta}}^{(EWLS)} = (\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\overline{\mathbf{y}}$ and $\hat{E}[y_i]^{(EWLS)} = \mathbf{x}_i'\hat{\boldsymbol{\beta}}^{(EWLS)}$

[8] See 4.2. Parametric Approach.

Step 4: Form the residuals vector $\mathbf{r} = \mathbf{y} - \hat{E}[\mathbf{y}]^{(EWLS)}$ form the estimated weighted least square (EWLS) fits in Step 3. Then perform LLR on the residuals $\mathbf{r}$ to get a smooth estimate $\hat{\mathbf{r}}$ of the mean of $\mathbf{r}$.

Step 5: Obtain the means model robust regression (MMRR) estimates via MRR2 as:

$$\hat{E}[\mathbf{y}]^{(MMRR)} = \hat{E}[\mathbf{y}]^{(EWLS)} + \lambda_{\mu}\hat{\mathbf{r}},$$

where $\lambda_{\mu} \in [0,1]$ is the asymptotically optimal means model mixing parameter with $\hat{y}^{(OLR)}$ replaced by $\hat{y}^{(EWLS)}$. For the nonparametric estimates, the bandwidths, b_{μ} and b_{σ}, are the values that minimize PRESS**[9]. Like the parametric and nonparametric approaches, once estimates of the mean and variance functions have been calculated, a squared error loss approach is used for process optimization. Furthermore, as in the nonparametric approach, the genetic algorithm is used for optimization because the estimates of the mean and variance functions do not take on closed form expressions. Pickle et al. (2006) used Box and Draper (1987) printing ink data to illustrate the performance of the semi-parametric method. We reanalyzed the data in example 4.1 by the parametric, nonparametric, and semi-parametric approaches. We found that the semi-parametric approach gives the smallest estimated squared error loss (SEL), and that the optimization based on the semi-parametric approach recommended settings resulted in the smallest estimated variance and the estimated process mean which is closest to the target value of 500.

[9] See 4.3. Nonparametric Approach.

Example 4.1. The printing ink example.

The Box and Draper (1987) printing ink study involved analyzing data with the RPD method. The objective was to examine the effects of three factors: speed (x_1), pressure (x_2), and distance(x_3), on the printing machine's ability to apply ink to package labels. This experiment was a 3^3 complete factorial design with three replicates at each design point. In this study, it was desired to determine an optimal setting was determined, at which process variance was the smallest and the process mean was nearest to the target of 500.

The results of the experiment are given in Table 4-1. Since two locations (i =10 and 14) have a sample standard deviation of zero, Pickle et al. (2006) replaced the observed sample variances, s_i^2 , with $s_i^2 + 1$ to fit the log transformation for the variance model. They assumed that the researcher had specified a first-order model for the log transformed variance model and a second-order model for the mean.

Table 4-1 Printing Ink Data

i	x_{1i}	x_{2i}	x_{3i}	y_{1i}	y_{2i}	y_{3i}	$\bar{y}_i$	s_i
1	-1	-1	-1	34	10	28	24	12.49
2	0	-1	-1	115	116	130	120.33	8.39
3	1	-1	-1	192	186	263	213.67	42.83
4	-1	0	-1	82	88	88	86	3.46
5	0	0	-1	44	178	188	136.67	80.41
6	1	0	-1	322	350	350	340.67	16.17
7	-1	1	-1	141	110	86	112.33	27257
8	0	1	-1	259	251	259	256.33	4.62
9	1	1	-1	290	280	245	271.67	23.63
10	-1	-1	0	81	81	81	81	0
11	0	-1	0	90	122	93	101.67	17.67
12	1	-1	0	319	376	376	357	32.91
13	-1	0	0	180	180	154	171.33	15.01
14	0	0	0	372	372	372	372	0
15	1	0	0	541	568	396	501.67	38.5
16	-1	1	0	288	192	312	264	63.5
17	0	1	0	432	336	513	427	88.61
18	1	1	0	713	725	754	730.67	21.08
19	-1	-1	1	364	99	199	220.67	133.82
20	0	-1	1	232	221	266	239.67	23.46
21	1	-1	1	408	415	443	422	18.52
22	-1	0	1	182	233	182	199	29.44
23	0	0	1	507	515	434	485.33	44.64
24	1	0	1	846	535	640	673.67	158.21
25	-1	1	1	236	426	468	476.67	55.51
26	0	1	1	660	440	403	501	438.94
27	1	1	1	878	991	1161	1010	142.45

For the nonparametric and semi-parametric approaches it was necessary to find an appropriate global bandwidth for the kernel function. Using PRESS**, we obtained a bandwidth

of 0.63 for the variance model and 0.52 for the means model. We also selected a bandwidth 0.51 for the nonparametric smoothing of the EWLS residuals in the semi-parametric fit to the mean.

In addition, for the semi-parametric approach we need to determine the appropriate mixing parameters, λ_σ and λ_μ. In the variance model, the asymptotically optimal data driven mixing parameter was 0.6812. This value suggests the presence of moderate amount of lack-of-fit in the parametric variance model. The asymptotically optimal data driven mixing parameter for the mean model was 1.0. This suggests that the presence of lack-of-fit for the parametric mean is severe. Adding the entire nonparametric residual fit is necessary corrections to correct the parametric means model.

By using the genetic algorithm with the squared error loss (SEL) objective function, we obtained the optimal factor settings. In Table 4-2, we see that x_1 are 1.00 for all three approach. x_2 value is also 1.00 in both the nonparametric and semi-parametric methods. However, x_3 coordinates are very different in all three approaches. Coincidentally, we found that x_3 is the most significant factor in the parametric variance model. In Table 4-2, we also see that the optimal factor settings for the semi-parametric approach (x_1=1.00, x_2 = 1.00, and x_3=-.532) provides the lowest estimated process variance of 1019.523 and the highest estimated process mean of 497.629 among the three methods. Also, the semi-parametric approach had the lowest estimated SEL value. Pickle et al. (2006) showed by a simulation study that the semi-parametric approach seems to perform better than the other methods in terms of variance and SEL for the setting considered by them

Table 4-2 Result

Approach	x_1	x_2	x_3	$\hat{E}[y_i]$	$\hat{Var}[y_i]$	$S\hat{E}L$
Parametric	1.000	0.358	-.112	497.619	1723.693	1729.363
Nonparametric	1.000	1.000	-.352	496.866	1088.455	1098.276
Semi-parametric	1.000	1.000	-.532	**497.629**	**1019.523**	**1025.150**

References And/Or Bibliography

Anderson-Cook, C.M., Prewitt, K., (2006), " Some guideline for using nonparametric methods for modeling data from response surface designs," *Journal of Modern Applied Statistical Methods* 4, 106-119

Borror, C.M. and Montgomery C.(1999), " Mixed Resolution Design as Alternatives To Taguchi Inner/Outer Array Designs for Robust Design Problems," *Quality and Reliability Engineering International*, 16, 117-127

Borror, C.M., Montgomery, D.C., and Myers, R.H.(2002), "Evaluation of statistical designs for Experiments with noise variables," *Journal of Quality Technology*, 34, 54-70.

Box, G., and Draper B. (1987), Empirical Model Building and Response Surface, Wiley, New York

Byrne, D. and Taguchi, G. (1987), "The Taguchi Approach to Parameter Design," *Quality Progress*, December, 1987, 19-26

Einsporn, R. and Birch, J., (1993), "Model robust regression: using nonparametric regression to improve parametric analyses," Technical report 93-5. Department of Statistics, Virginia Tech, Blacksburg, VA.

Jones, S., and Box G., (1990), "Robust product designs, Part I: First-order models with design x environment interactions," *Center for Quality and Productivity Improvement*, 62.

Jones, S., and Box G., (1990), "Robust product designs, Part II: Second-order models," *Center for Quality and Productivity Improvement*, 63.

Jones, S., and Box G., (1990), "Robust product designs, Part III: Second-order models with additional third-order terms," *Center for Quality and Productivity Improvement*, 64.

Mays, J., Birch, J. and Starnes, B. (2001), "Model robust regression : combining parametric, nonparametric and semiparametric methods," *Journal of Nonparametric Statistics*, 13, 245-277

Montgomery, D.C. (2001), Design and Analysis of Experiments, 5th edition, John Wiley & Sons, New York

Myers, R.H., Khuri, A.I., and Vinning, G (1992), "Response surface alternatives to the Taguchi robust parameter design approach," *The American Statistician*, 462, 131-139.

Myers, R.H., Kim, Y., and Griffiths, K. (1997), "Response surface methods and use of noise variables," *Journal of Quality Technology*, 29, 429-441

Nair, S. N.(1992), " Taguchi's Parameter Design: A Panel Discussion," Technology,34, 127-159

Pickle, S.M., Robinson, T.J., Birch, J.B., and Anderson-Cook, C.M.(2005), "Robust Parameter design : A semi-parametric approach," Technical report No.05-7, Department of Statistics, Virginia Tech, Blacksburg, VA

Pignatiello J.(1988), "An Overview of the Strategy and Tactics of Taguchi," *IIE Transaction*, 30, 247-254

Robinson, T. and Birch, J. (2002), "Dual model robust regression: robust to model misspecification," Technical report 02-2. Department of Statistics, Virginia Tech, Blacksburg, VA.

Shoemaker A.C., Tsui K. and Wu C.(1991), " Economical Experimentation Methods for Robust Parameter Design," *IIQP Research Report*, RR-89-04

Vinning, G.G., and Bohn, L. (1998), "Response surfaces for the mean and the process variance using a nonparametric approach," *Journal of Quality Technology*, 30, 282-291.

Vining, G.G., Myers, R.H.(1990),"Combining Taguchi and response surface philosophies: a dual response approach," *Journal of Quality Technology*, 22,38-45

Appendix-Computer Codes

1. Example 3.1- The pilot plant experiment

-SAS code

```
data a;
input z1       x1      x2      x3      y;
cards;
-1      -1      -1      -1      45
1       -1      -1      -1      71
-1      1       -1      -1      48
1       1       -1      -1      65
-1      -1      1       -1      68
1       -1      1       -1      60
-1      1       1       -1      80
1       1       1       -1      65
-1      -1      -1      1       43
1       -1      -1      1       100
-1      1       -1      1       45
1       1       -1      1       104
-1      -1      1       1       75
1       -1      1       1       86
-1      1       1       1       70
1       1       1       1       96
;
proc rsreg;
model y= z1 x1 x2 x3;
run;
```

- R code

```
x2=seq(-1,1,0.5)
x3=seq(-1,1,0.5)

model=function(a,b){sqrt(136.42+82.08*a*a+69.06*b*b-195.88*a+179.66*b-150.58*a*b)}
z=outer(x2,x3,model)

model1=function(a,b){70.06+4.94*a+7.31*b}
z1=outer(x2,x3,model1)

contour(x2,x3,z1,labcex = 2) /*Figure 3-1 Contour of mean filtration rate*/
contour(x2,x3,z,labcex = 2) /*Figure 3-2 Contour plot of propagation of error (a) contour plot*/

persp(x2,x3,z,theta=50,phi=10,ticktype="detailed",col = "lightblue")
persp(x2,x3,z,theta=485,phi=350,ticktype="detailed",col = "lightblue")
/*Figure 3-2 Contour plot of propagation of error (b) response surface plot*/
```

```
contour(x2,x3,z,labcex = 2)
contour(x2,x3,z1,method = "simple",labcex = 2,lty = 2,add=TRUE,col="Red")
/*Figure 3-3 overlay plot of the contours of the mean filtration rate and the POE*/
```

2. Example 4.1- The Printing ink example

Text file- coded.txt

```
x1,x2,x3,ybar,s,t2
0,0,0,24.0000,12.489996,5.056246
0.5,0,0,120.3333,8.386497,4.267364
1,0,0,213.6667,42.829118,7.514981
0,0.5,0,86.0000,3.464102,2.564949
0.5,0.5,0,136.6667,80.407297,8.774365
1,0.5,0,340.6667,16.165808,5.569616
0,1,0,112.3333,27.574142,6.635071
0.5,1,0,256.3333,4.618802,3.106080
1,1,0,271.6667,23.629078,6.326746
0,0,0.5,81.0000,0.000000,0.000000
0.5,0,0.5,101.6667,17.672955,5.747268
1,0,0.5,357.0000,32.908965,6.988413
0,0.5,0.5,171.3333,15.011107,5.422009
0.5,0.5,0.5,372.0000,0.000000,0.000000
1,0.5,0.5,501.6667,92.500450,9.054544
0,1,0.5,264.0000,63.498031,8.302266
0.5,1,0.5,427.0000,88.605869,8.968524
1,1,0.5,730.6667,21.079216,6.098823
0,0,1,220.6667,133.822021,9.793077
0.5,0,1,239.6667,23.459184,6.312340
1,0,1,422.0000,18.520259,5.840642
0,0.5,1,199.0000,29.444864,6.766192
0.5,0.5,1,485.3333,44.635561,7.597564
1,0.5,1,673.6667,158.209776,10.127884
0,1,1,176.6667,55.509759,8.033442
0.5,1,1,501.0000,138.935237,9.868068
1,1,1,1010.0000,142.453501,9.918081
```

-R code (This R code comes from Virginia Tech Department of Statistics TECHNICAL REPORT NO. 05-7)

```
ink.data.coded<-read.table('g:\coded.txt',sep=",",header=TRUE)
x1<-ink.data.coded[,1]                    #x1=speed
x2<-ink.data.coded[,2]                    #x2=pressure
x3<-ink.data.coded[,3]                    #x3=distance
X.design<-cbind(x1,x2,x3)          #design matrix
ybar<-ink.data.coded[,4]           #mean response
s<-ink.data.coded[,5]              #standard deviation
t<-ink.data.coded[,6]              #transformation of standard deviation = log(s^2+1)
```

```
#####PARAMETRIC APPROACH#####

#The following code will estimate responses parametrically using OLS for the transformed variance and EWLS for the mean#

#Preliminary Functions#

#Leave one out Cross-validation
yhat.minusi.p.fn<-function(y,res,H){
        n<-length(y)
        h<-diag(H)          #grabs diagonals from H
        yhat.minusi<-rep(0,n)
        for(i in 1:n){
                yhat.minusi[i]<-y[i]-(res[i]/(1-h[i]))
        }
        return(yhat.minusi)
}
obj.p.fn<-function(beta.mean,beta.t,theta,x0){
        #This function will find the estimated MSE=(yhat-theta)^2+varhat
        #at the location x0 based on the estimated ols functions for the
        #transformed variance and wls functions for mean
        #here the mean model is second order
        #and the t model is first order
        x1.x2<-x0[1]*x0[2]
        x1.x3<-x0[1]*x0[3]
        x2.x3<-x0[2]*x0[3]
        x1.x1<-x0[1]*x0[1]
        x2.x2<-x0[2]*x0[2]
        x3.x3<-x0[3]*x0[3]
        modelx0.mean<-c(1,x0,x1.x2,x1.x3,x2.x3,x1.x1,x2.x2,x3.x3)
        modelx0.t<-c(1,x0)
        yhat<-beta.mean%*%modelx0.mean
        bias<-yhat-theta
        that<-beta.t%*%modelx0.t
        varhat<-exp(that)-1
        msehat<-bias^2+varhat
        msehat
}
ga.p.fn<-function(beta.mean,beta.t,theta){
        #This function will perform a Simple Genetic Algorithm
        #For the objective function estimated MSE=(yhat-theta)^2+var
        #Where yhat is the estimate of the mean and theta is the target value
        #And var is the estimate of the variance
        #var will be found with ols and ybar will be found with wls

        #Stopping Criteria
        maxiter<-10000                          #set maximum number of iterations
        satiter<-1000       #set number of iterations during which the best results keep saturating
```

```
epsln<-1e-8                                  #smallest gain worth recognizing

#GA Parameters
m<-50                                        #population size at each generation
mutrate<-.2                                  #mutation rate
crossrate<-.9                                #crossover rate
zerorate<-.2                                 #zero rate
extremerate<-.2                              #extreme rate
keep<-2                          #keep top two from each generation to remain unchanged
num.parents<-m-keep              #number of population that will be used for mating
mate<-ceiling((num.parents)/2)   #number of matings

#Create initial population and initialize 'best result' holders
xrange<-matrix(c(0,1,0,1,0,1),2,3)
k<-ncol(xrange)
elite<-matrix(0,keep,k)                      #matrix for elite chromosomes to remain unchanged
cross<-matrix(0,num.parents,k)   #matrix for crossover
iter<-0                                      #generation (iteration) counter
stopcode<-0
inarow<-0 #number of iterations with function's value consecutively less than epsln
bestfun<-1e30                                #essential positive infinity
bestx<-matrix(0,1,k)
f<-rep(0,m)                                  #initialize function value for each of m vectors
G<-matrix(runif(m*k,0,1),m,k)    #initial generation with population size m

#####Iterate through generations#####
while(stopcode==0){
iter<-iter+1                     #increments counter
if(iter>maxiter) stopcode<-2     #loop will exit on stopcode=2 for exceeding the maximum number of iterations

        #Evaluate current generation
        for(i in 1:m){
                f[i]<-obj.p.fn(beta.mean,beta.t,theta,G[i,])
                }

        mat<-cbind(G,f)#create matrix of x locations and corresponding function values
        newmat<-mat[order(mat[,4]),]     #sort matrix of by function values increasing
        minf<-newmat[1,4]                #optimal function (minimum)
        bf<-min(minf, bestfun)
        fgain<-bestfun-bf       #fgain is always non-negative it measure the change
        if(fgain>epsln)         inarow<-0 else inarow<-(inarow+1)

        if(fgain>0){
                bestfun<-bf
                bestx<-newmat[1,1:3]             #optimal x location
                }
```

```
if(inarow>satiter) stopcode<-1#loop will exit on stopcode=1 for best result having been achieved

                #Select elite to remain unchanged
                elite<-newmat[1:keep,1:3]
                #Select parents for mating
                cross<-newmat[(keep+1):m,1:3]

                #Do Crossover
                for(i in 1:mate){
                        z<-rbinom(1,1,crossrate)
                        if(z==1){
                                x<-ceiling(runif(1,0,(k-1)))
                                temp<-cross[i,(x+1):k]
                                cross[i,(x+1):k]<-cross[i+mate,(x+1):k]
                                cross[i+mate,(x+1):k]<-temp
                                }
                        }

                #Do Mutation
                M<-matrix(runif((num.parents*k),0,1),num.parents,k)
                for(i in 1:num.parents){
                        for(j in 1:k){
                                zz<-rbinom(1,1,mutrate)
                                if(zz==1) cross[i,j]<-M[i,j]
                                }
                        }
                #Do Zero Gene Operator
                for(i in 1:num.parents){
                        for(j in 1:k){
                                zzz<-rbinom(1,1,zerorate)
                                if(zzz==1) cross[i,j]<-0
                                }
                        }
                #Do Extreme Values Operator
                Extreme<-matrix(1,num.parents,k)
                for(i in 1:num.parents){
                        for(j in 1:k){
                                zzzz<-rbinom(1,1,extremerate)
                                if(zzzz==1) cross[i,j]<-Extreme[i,j]
                                }
                        }
                G<-rbind(elite,cross)
        }

        if(iter<maxiter) status<-0
        else status<-1
```

```
        result<-c(bestx,bestfun,iter)
        return(result)
}


#Create model matrices#

ones<-rep(1,length(x1))                                 #intercept column
X.model.1st<-cbind(ones,X.design)           #first-order model matrix

#TwoWay Interactions
x1.x2<-x1*x2
x1.x3<-x1*x3
x2.x3<-x2*x3

#Quadratic Terms
x1.x1<-x1^2
x2.x2<-x2^2
x3.x3<-x3^2

X.model.2nd<-cbind(X.model.1st,x1.x2,x1.x3,x2.x3,x1.x1,x2.x2,x3.x3)         #second-order model matrix

#FIRST-ORDER OLS FOR TRANSFORMED VARIANCE#

t.ols<-lm(t~x1+x2+x3)
betas.t.ols<-t.ols$coefficients
betas.t.ols
fit.t.ols<-t.ols$fitted.values
fit.t.ols
res.t.ols<-t.ols$residuals
res.t.ols
sum.t.ols<-summary(t.ols)
sum.t.ols
anova(t.ols)
par(mfrow=c(2,2))
plot(t.ols,pch=16)
mtext("Residual Plots for First-Order Model For Variance Transformation Using OLS",outer=T,line=-2)

#Hat matrix for OLS
H.t.ols<-X.model.1st%*%(solve(t(X.model.1st)%*%X.model.1st))%*%t(X.model.1st)
H.t.ols

#Get variance weights for means model
varhat.ols<-exp(fit.t.ols)-1
varhat.ols
wts.t.ols<-1/varhat.ols
wts.t.ols
```

```
W.t.ols<-diag(wts.t.ols)

#Leave one out Cross-validation
fit.minusi.t.ols<-yhat.minusi.p.fn(t,res.t.ols,H.t.ols)
fit.minusi.t.ols
res.minusi.t.ols<-t-fit.minusi.t.ols
res.minusi.t.ols

#SECOND-ORDER EWLS FOR MEAN#

mean.ewls<-lm(ybar~x1+x2+x3+x1.x2+x1.x3+x2.x3+x1.x1+x2.x2+x3.x3,weights=wts.t.ols)
betas.mean.ewls<-mean.ewls$coefficients
betas.mean.ewls
fit.mean.ewls<-mean.ewls$fitted.values
fit.mean.ewls
res.mean.ewls<-mean.ewls$residuals
res.mean.ewls
sum.mean.ewls<-summary(mean.ewls)
sum.mean.ewls
anova(mean.ewls)
par(mfrow=c(2,2))
plot(mean.ewls,pch=16)
mtext("Residual Plots for Second-Order Model For Mean Using EWLS",outer=T,line=-2)

#Hat matrix for EWLS
H.mean.ewls<-X.model.2nd%*%(solve(t(X.model.2nd)%*%W.t.ols%*%X.model.2nd))%*%t(X.model.2nd)%*%W.t.ols
H.mean.ewls
#OPTIMIZE WITH GA#
#Mean target is 500
ga.p.fn(betas.mean.ewls,betas.t.ols,500)
#NONPARAMETRIC APPROACH#

#The following code will estimate responses nonparametrically using LLR for the transformed variance and EWLLR for the mean#

#Preliminary Functions#

#Get Kernel Weights at x0
kern.fn<-function(X,y,b,x0){
        #This function will find the normal kernel weights at the location x0
        #Given the design matrix X, the observed values y, and the bandwidth b
        n<-length(y)                    #number of obervations
        p<-ncol(X)                      #number of regressors
        u<-matrix(0,n,p)#matrix of arguement for kernel function with n rows and p columns
        kern.norm<-matrix(0,n,p)        #matrix for kernel function with n rows and p columns
        K.norm<-rep(0,n)                #vector of mult. norm. kernel
        wt<-rep(0,n)                    #vector of kernel weights with n rows
        for(j in 1:p){
```

```
                for(i in 1:n){
                        u[i,j]<-(x0[j]-X[i,j])/b
                        kern.norm[i,j]<-exp(-(u[i,j]^2))   #normal kernel
                }
        }
        K.norm<-kern.norm[,1]*kern.norm[,2]*kern.norm[,3]   #multiplicative normal kernel for p=3 regressors
        for(i in 1:n){
                        wt[i]<-K.norm[i]/sum(K.norm)                #kernel weights
        }
        return(wt)
}
#Get LPR Estimate at x0
lpr.fn<-function(X,y,b,d,x0){
        #This function will find the Local Polynomial Regression estimate at the location x0
        #Using the normal kernel for the weights
        #Given the design matrix X, the observed values y, the bandwidth b, and the degree d
        #If d = 0 a constant will be fit; i.e., Kernel Regression
        #If d = 1 a simple linear regression line will be fit; i.e., Local Linear Regression
        wt<-kern.fn(X,y,b,x0)#kernel weight vector
        yhat<-0
        if(d==0){
                model<-lm(y~1,weight=wt)
                beta<-model$coefficients
                beta0<-beta[1]
                beta1<-0
                beta2<-0
                beta3<-0
        }else{model<-lm(y~X,weight=wt)
                beta<-model$coefficients
                beta0<-beta[1]
                beta1<-beta[2]
                beta2<-beta[3]
                beta3<-beta[4]
        }

        yhat<-beta0+beta1*x0[1]+beta2*x0[2]+beta3*x0[3]
        return(yhat)
}
#Get Weighted LPR Estimate at x0
lpr.w.fn<-function(X,y,b,d,x0,W){
#This function will find the Weigthed Local Polynomial Regression estimate at the location x0
        #Using the normal kernel for the weights
        #Given the design matrix X, the observed values y, the bandwidth b, and the degree d
        #and the estimated weight matrix from variance model W
        #If d = 0 a constant will be fit; i.e., Kernel Regression
        #If d = 1 a simple linear regression line will be fit; i.e., Local Linear Regression
        k<-kern.fn(X,y,b,x0)  #kernel weight vector
```

```
        k.half<-sqrt(k)
        K.half<-diag(k.half)
        W.star<-K.half%*%W%*%K.half
        wt<-diag(W.star)
        yhat<-0
        if(d==0){
                model<-lm(y~1,weight=wt)
                beta<-model$coefficients
                beta0<-beta[1]
                beta1<-0
                beta2<-0
                beta3<-0
        }else{model<-lm(y~X,weight=wt)
                beta<-model$coefficients
                beta0<-beta[1]
                beta1<-beta[2]
                beta2<-beta[3]
                beta3<-beta[4]
        }
        yhat<-beta0+beta1*x0[1]+beta2*x0[2]+beta3*x0[3]
        return(yhat)
}
#Fit original data points with LPR
fit.lpr.fn<-function(X,y,b,d){
        #This function will fit the original data points by local polynomial regression with a degree (d) polynomial
        #Given the design matrix X, the observed y values, and the bandwidth b
        n<-length(y)            #number of observations
        X0<-X                   #fit at original data points
        fit<-rep(0,n)
        for(i in 1:n){
                fit[i]<-lpr.fn(X,y,b,d,X0[i,])
        }
        return(fit)
}
#Fit original data points with weighted LPR
fit.lpr.w.fn<-function(X,y,b,d,W){
        #This function will fit the original data points by local polynomial regression with a degree (d) polynomial
        #Given the design matrix X, the observed y values, and the bandwidth b
        #and the estimated weight matrix from variance model W
        n<-length(y)            #number of observations
        X0<-X                   #fit at original data points
        fit<-rep(0,n)
        for(i in 1:n){
                fit[i]<-lpr.w.fn(X,y,b,d,X0[i,],W)
        }
        return(fit)
}
```

```
#Leave one out Cross-validation
fit.lpr.minusi.fn<-function(X,y,b,d){
	#This function will fit the original data points using minus i technique for LPR
	#Given the design matrix X, the observed values y, the bandwidth b, the degree of polynomial d
	n<-length(y)
	X0<-X
	fit.minusi<-rep(0,n)
	for(i in 1:n){
		fit.minusi[i]<-lpr.fn(X[-i,],y[-i],b,d,X0[i,])
	}
	return(fit.minusi)
}
#Weighted Leave one out Cross-validation
fit.lpr.w.minusi.fn<-function(X,y,b,d,wt){
	#This function will fit the original data points using minus i technique for Weigthed LPR
	#Given the design matrix X, the observed values y, the bandwidth b, the degree of polynomial d
	#and the estimated weight matrix from variance model W
	num<-length(y)
	X0<-X
	fit.w.minusi<-rep(0,num)
	W.minusi<-array(0,dim=c(num-1,num-1,num))
	for(i in 1:num){
		W.minusi[,,i]<-diag(wt[-i])
	}
	for(i in 1:num){
		fit.w.minusi[i]<-lpr.w.fn(X[-i,],y[-i],b,d,X0[i,],W.minusi[,,i])
	}
	return(fit.w.minusi)
}
#Get row of Hat Matrix for x0
Hrow.fn<-function(X,y,b,d,x0){
	#This function will find the row of the Hat matrix corresponding to x0 such that Hrow*y=yhat at x0
	#Given the design matrix X, the observed values y, the bandwidth b, the degree d of the LPR estimate
	#If d = 0 kernel regression
	#If d = 1 LLR
	n<-length(y)				#number of observations
	k<-ncol(X)				#number of regressors
	p<-k+1					#number of parameters
	ones<-rep(1,n)
	modelX<-cbind(ones,X)
	modelx0<-c(1,x0)
	Hrow<-rep(0,n)
	wt.kern<-kern.fn(X,y,b,x0)		#vector of kernel weights with n rows
	kern.mat<-diag(wt.kern)			#diagonal matrix of kernel weights
	XpWX<-t(modelX)%*%kern.mat%*%modelX
	inv<-solve(XpWX)
	if(d==0){
```

```
                Hrow<-wt.kern
                }else{
                        Hrow<-modelx0%*%inv%*%t(modelX)%*%kern.mat
                }
        return(Hrow)
}
#Get row of weighted Hat Matrix for x0
Hrow.w.fn<-function(X,y,b,d,x0,W){
        #This function will find the row of the Hat matrix corresponding to x0 such that Hrow*y=yhat at x0
        #Given the design matrix X, the observed values y, the bandwidth b, the degree d of the LPR estimate
        #and the estimated weight matrix from variance model W
        #If d = 0 kernel regression
        #If d = 1 LLR
        n<-length(y)                            #number of observations
        k<-ncol(X)                              #number of regressors
        p<-k+1                                  #number of parameters
        ones<-rep(1,n)
        modelX<-cbind(ones,X)
        modelx0<-c(1,x0)
        Hrow.w<-rep(0,n)
        wt.kern<-kern.fn(X,y,b,x0)      #vector of kernel weights with n rows
        wt.kern.half<-sqrt(wt.kern)
        kern.half.mat<-diag(wt.kern.half)#diagonal matrix of half kernel weights
        W.mat<-kern.half.mat%*%W%*%kern.half.mat        #weight matrix
        XpWX<-t(modelX)%*%W.mat%*%modelX
        inv<-solve(XpWX)
        if(d==0){
                Hrow.w<-diag(W.mat)
                }else{
                        Hrow.w<-modelx0%*%inv%*%t(modelX)%*%W.mat
                }
        return(Hrow.w)
}
#Hat matrix for lpr
H.lpr.fn<-function(X,y,b,d){
        n<-length(y)                    #number of observations
        X0<-X                           #fit at original data points
        H<-matrix(0,n,n)        #create H matrix
        for(i in 1:n){
                H[i,]<-Hrow.fn(X,y,b,d,X0[i,])
        }
        return(H)
}
#Hat matrix for weighted lpr
H.lpr.w.fn<-function(X,y,b,d,W){
        n<-length(y)                    #number of observations
        X0<-X                           #fit at original data points
```

```
	H.w<-matrix(0,n,n)    #create H matrix
	for(i in 1:n){
		H.w[i,]<-Hrow.w.fn(X,y,b,d,X0[i,],W)
	}
	return(H.w)
}
#Get Trace of square matrix
trace.fn<-function(X){
	#This function will find the trace of a square matrix
	n<-nrow(X)
	diag<-rep(0,n)
	for(i in 1:n){
		diag[i]<-X[i,i]
	}
	trace<-sum(diag)
	return(trace)
}
#Get SSE
sse.fn<-function(y,yhat){
	#This function will find SSE given observed values y and fitted values yhat
	res<-y-yhat
	sse<-t(res)%*%res
	return(sse)
}
#Get weighted SSE
sse.w.fn<-function(y,yhat,W){
	#This function will find weighted SSE given observed values y, fitted values yhat
	#and the estimated weight matrix from variance model W
	res<-y-yhat
	sse.w<-t(res)%*%W%*%res
	return(sse.w)
}
#Get PRESS** for one bandwidth value
press_star_star.b.fn<-function(X,y,b,d){
	#This function will find the PRESS** value corresponding to a given bandwidth value(b)
	#Given the design matrix X, the observed values y, and the degree (d) of the polynomial for LPR
	n<-length(y)                    #number of observations
	k<-ncol(X)                      #number of regressors
	p<-k+1                          #number of parameters
	X0<-X                           #fit at original data points
	H<-matrix(0,n,n)                #create H matrix
	Htrace<-0                       #create variable for trace of H values
	yhat<-rep(0,n)                  #create vector for yhat values
	press<-rep(0,n)                 #create vector for PRESS values
	press_star_star<-0              #create variable for PRESS** value
	for(i in 1:n){
		H[i,]<-Hrow.fn(X,y,b,d,X0[i,])
```

```
	}
	Htrace<-trace.fn(H)
	yhat.minusi<-fit.lpr.minusi.fn(X,y,b,d)
	yhat<-fit.lpr.fn(X,y,b,d)
	res<-y-yhat
	SSEb<-t(res)%*%res
	if(d==0){
		MSE<-(summary(lm(y~1))$sigma)^2
		dfE<-summary(lm(y~1))$df[2]
		SSEmax<-MSE*dfE
	}else{MSE<-(summary(lm(y~X))$sigma)^2
		dfE<-summary(lm(y~X))$df[2]
		SSEmax<-MSE*dfE
	}
	for(i in 1:n){
		press[i]<-(y[i]-yhat.minusi[i])^2
	}
	press_star_star<-sum(press)/(n-Htrace+(n-p)*((SSEmax-SSEb)/SSEmax))
	return(press_star_star)
}
#Get weighted PRESS** for one bandwidth value
press_star_star.b.w.fn<-function(X,y,b,d,wt){
	#This function will find the PRESS** value corresponding to a given bandwidth value(b)
	#Given the design matrix X, the observed values y, and the degree (d) of the polynomial for LPR
	#and the estimated weights wt
	n<-length(y)				#number of observations
	k<-ncol(X)				#number of regressors
	p<-k+1					#number of parameters
	X0<-X					#fit at original data points
	H.w<-matrix(0,n,n)		#create H matrix
	Htrace<-0			#create variable for trace of H values
	yhat<-rep(0,n)				#create vector for yhat values
	press.w<-rep(0,n)		#create vector for PRESS values
	press_star_star.w<-0		#create variable for PRESS** value
	W<-diag(wt)				#Create weight matrix
	for(i in 1:n){
		H.w[i,]<-Hrow.w.fn(X,y,b,d,X0[i,],W)
	}
	Htrace<-trace.fn(H.w)
	yhat.w<-fit.lpr.w.fn(X,y,b,d,W)
	res.w<-y-yhat.w
	SSEb.w<-t(res.w)%*%W%*%res.w
	yhat.minusi.w<-fit.lpr.w.minusi.fn(X,y,b,d,wt)
	if(d==0){
		MSE<-(summary(lm(y~1,weights=wt))$sigma)^2
		dfE<-summary(lm(y~1,weights=wt))$df[2]
		SSEmax<-MSE*dfE
```

```
	}else{MSE<-(summary(lm(y~X,weights=wt))$sigma)^2
		dfE<-summary(lm(y~X,weights=wt))$df[2]
		SSEmax<-MSE*dfE
	}
	for(i in 1:n){
		press.w[i]<-((y[i]-yhat.minusi.w[i])^2)*wt[i]
	}
	press_star_star.w<-sum(press.w)/(n-Htrace+(n-p)*((SSEmax-SSEb.w)/SSEmax))
	return(press_star_star.w)
}
#Get Optimal Bandwidth from Candidate List use with coded x's (0,1)
band.press_star_star.fn<-function(X,y,band,d){
	#This function will find the optimal bandwidth that minimizes PRESS**
	#given a candidate list of bandwidth values (band)
	m<-length(band)
	press_star_star.value<-rep(1e10,m)
	tol<-rep(0,m)
	press_star_star.value[1]<-press_star_star.b.fn(X,y,band[1],d)
	limit<-0.01
	for(i in 2:m){
		press_star_star.value[i]<-press_star_star.b.fn(X,y,band[i],d)
		tol[i]<-abs(press_star_star.value[i]-press_star_star.value[i-1])/press_star_star.value[i-1]
		if(tol[i]<=limit){break}
	}
	mat<-cbind(band,press_star_star.value,tol)   #create matrix of all b and corresponding PRESS** values
	newmat<-mat[order(mat[,2]),]                 #sort matrix of b and PRESS** values by PRESS** increasing
	bopt<-newmat[1,1]                            #optimal bandwidth
	press_star_star_opt<-newmat[1,2]             #corresponding optimal PRESS** (minimum)
	return(bopt,press_star_star_opt)
}
#Get weighted PRESS** Optimal Bandwidth from Candidate List use with coded x's (0,1)
band.press_star_star.w.fn<-function(X,y,band,d,wt){
	#This function will find the optimal bandwidth that minimizes PRESS**
	#given a candidate list of bandwidth values (band)
	#and the estimated weights wt
	m<-length(band)
	press_star_star.w.value<-rep(1e10,m)
	tol<-rep(0,m)
	press_star_star.w.value[1]<-press_star_star.b.w.fn(X,y,band[1],d,wt)
	limit<-0.01
	for(i in 2:m){
		press_star_star.w.value[i]<-press_star_star.b.w.fn(X,y,band[i],d,wt)
		tol[i]<-abs(press_star_star.w.value[i]-press_star_star.w.value[i-1])/press_star_star.w.value[i-1]
		if(tol[i]<=limit){break}
	}
	mat<-cbind(band,press_star_star.w.value,tol)          #create matrix of all b and corresponding PRESS** values
	newmat<-mat[order(mat[,2]),]                 #sort matrix of b and PRESS** values by PRESS** increasing
```

```
	bopt<-newmat[1,1]			#optimal bandwidth
	press_star_star_opt<-newmat[1,2]	#corresponding optimal PRESS** (minimum)
	return(bopt,press_star_star_opt)
}
#Get Plot of PRESS** vs Bandwidth from Candidate List
band.press_star_star.plot.fn<-function(X,y,band,d){
	#This function plot PRESS** values vs bandwidth values
	#given a candidate list of bandwidth values (band)
	m<-length(band)
	press_star_star.value<-rep(0,m)
	for(i in 1:m){
		press_star_star.value[i]<-press_star_star.b.fn(X,y,band[i],d)
	}
mat<-cbind(band,press_star_star.value)#create matrix of all b and corresponding PRESS** values
newmat<-mat[order(mat[,2]),]#sort matrix of b and PRESS** values by PRESS** increasing
	bopt<-newmat[1,1]			#optimal bandwidth
	press_star_star_opt<-newmat[1,2]	#corresponding optimal PRESS** (minimum)
	plot(mat[,1],mat[,2],xlab="Bandwidth",ylab="PRESS**",type="l")
}
#Get Plot of Weighted PRESS** vs Bandwidth from Candidate List
band.press_star_star.plot.w.fn<-function(X,y,band,d,wt){
	#This function plot PRESS** values vs bandwidth values
	#given a candidate list of bandwidth values (band)
	#and the estimated weights wt
	m<-length(band)
	press_star_star.w.value<-rep(0,m)
	for(i in 1:m){
		press_star_star.w.value[i]<-press_star_star.b.w.fn(X,y,band[i],d,wt)
	}
mat<-cbind(band,press_star_star.w.value)#create matrix of all b and corresponding PRESS** values
newmat<-mat[order(mat[,2]),]#sort matrix of b and PRESS** values by PRESS** increasing
	bopt<-newmat[1,1]			#optimal bandwidth
	press_star_star_opt<-newmat[1,2]	#corresponding optimal PRESS** (minimum)
	plot(mat[,1],mat[,2],xlab="Bandwidth",ylab="PRESS**",type="l")
}
obj.llr.fn<-function(X,y,theta,b1,d1,t,b2,d2,x0,W){
	#This function will find the estimated MSE=(yhat-theta)^2+varhat
	#Given the data (X), the mean response (y), the target value for
	#the mean (theta), the bandwidth for the estimation of the mean (b1), the
	#degree of the lpr for the mean (d1), the transformed variance (t),
	#the bandwidth for the estimation of the transformed variance (b2), the
	#degree of the lpr for the transformed variance, and the location of
	#estimation (x0)

	yhat<-lpr.w.fn(X,y,b1,d1,x0,W)
	bias<-yhat-theta
	that<-lpr.fn(X,t,b2,d2,x0)
```

```
        varhat<-exp(that)-1
        msehat<-bias^2+varhat
        msehat
}

ga.llr.fn<-function(X,y,theta,b1,d1,t,b2,d2,W){
        #This function will perform a Simple Genetic Algorithm
        #For the objective function estimated MSE=(yhat-theta)^2+var
        #Given the data (X), the mean response (y), the target value for
        #the mean (theta), the bandwidth for the estimation of the mean (b1), the
        #degree of the lpr for the mean (d1), the transformed variance (t),
        #the bandwidth for the estimation of the transformed variance (b2), the
        #degree of the lpr for the transformed variance

        #Stopping Criteria
        maxiter<-10000                  #set maximum number of iterations
        satiter<-1000                   #set number of iterations during which the best results keep saturating
        epsln<-1e-8                     #smallest gain worth recognizing

        #GA Parameters
        m<-50                           #population size at each generation
        mutrate<-.2                     #mutation rate
        crossrate<-.9                   #crossover rate
        zerorate<-.2                    #zero rate
        extremerate<-.2                 #extreme rate
        keep<-2                         #keep top two from each generation to remain unchanged
        num.parents<-m-keep             #number of population that will be used for mating
        mate<-ceiling((num.parents)/2)  #number of matings

        #Create initial population and initialize 'best result' holders
        xrange<-matrix(c(0,1,0,1,0,1),2,3)
        k<-ncol(xrange)
        elite<-matrix(0,keep,k)         #matrix for elite chromosomes to remain unchanged
        cross<-matrix(0,num.parents,k)  #matrix for crossover
        iter<-0                         #generation (iteration) counter
        stopcode<-0
        inarow<-0                       #number of iterations with function's value consecutively less than epsln
        bestfun<-1e30                   #essential positive infinity
        bestx<-matrix(0,1,k)
        f<-rep(0,m)                     #initialize function value for each of m vectors
        G<-matrix(runif(m*k,0,1),m,k)   #initial generation with population size m

        #####Iterate through generations#####
        while(stopcode==0){
                iter<-iter+1            #increments counter
                if(iter>maxiter) stopcode<-2    #loop will exit on stopcode=2 for exceeding the maximum number of iterations
```

```
#Evaluate current generation
for(i in 1:m){
        f[i]<-obj.llr.fn(X,y,theta,b1,d1,t,b2,d2,G[i,],W)
        }

mat<-cbind(G,f)                         #create matrix of x locations and corresponding function values
newmat<-mat[order(mat[,4]),]    #sort matrix of by function values increasing
minf<-newmat[1,4]               #optimal function (minimum)
bf<-min(minf, bestfun)
fgain<-bestfun-bf       #fgain is always non-negative it measure the change
if(fgain>epsln)         inarow<-0 else inarow<-(inarow+1)

if(fgain>0){
        bestfun<-bf
        bestx<-newmat[1,1:3]            #optimal x location
        }

if(inarow>satiter) stopcode<-1    #loop will exit on stopcode=1 for best result having been achieved

#Select elite to remain unchanged
elite<-newmat[1:keep,1:3]

#Select parents for mating
cross<-newmat[(keep+1):m,1:3]

#Do Crossover
for(i in 1:mate){
        z<-rbinom(1,1,crossrate)
        if(z==1){
                x<-ceiling(runif(1,0,(k-1)))
                temp<-cross[i,(x+1):k]
                cross[i,(x+1):k]<-cross[i+mate,(x+1):k]
                cross[i+mate,(x+1):k]<-temp
                }
        }

#Do Mutation
M<-matrix(runif((num.parents*k),0,1),num.parents,k)
for(i in 1:num.parents){
        for(j in 1:k){
                zz<-rbinom(1,1,mutrate)
                if(zz==1) cross[i,j]<-M[i,j]
                }
        }
#Do Zero Gene Operator
for(i in 1:num.parents){
        for(j in 1:k){
```

```
                                        zzz<-rbinom(1,1,zerorate)
                                        if(zzz==1) cross[i,j]<-0
                                        }
                                }
                        #Do Extreme Values Operator
                        Extreme<-matrix(1,num.parents,k)
                        for(i in 1:num.parents){
                                for(j in 1:k){
                                        zzzz<-rbinom(1,1,extremerate)
                                        if(zzzz==1) cross[i,j]<-Extreme[i,j]
                                        }
                                }
                        G<-rbind(elite,cross)
                }

                if(iter<maxiter) status<-0
                else status<-1

                result<-c(bestx,bestfun,iter)
                return(result)
}

#LLR FOR TRANSFORMED VARIANCE#
#Find optimal bandwidth from candidate list based on coded x's
par(mfrow=c(1,1))
band<-seq(.1,1,.01)
band.press_star_star.plot.fn(X.design,t,band,1)
band<-seq(.3,1,.01)
bopt_t<-band.press_star_star.fn(X.design,t,band,1)$bopt
bopt_t

#Find LLR Estimates for transformed variance
fit.t.llr<-fit.lpr.fn(X.design,t,bopt_t,1)
fit.t.llr
res.t.llr<-t-fit.t.llr
res.t.llr

#Leave one out Cross-validation
fit.minusi.t.llr<-fit.lpr.minusi.fn(X.design,t,bopt_t,1)
fit.minusi.t.llr
res.minusi.t.llr<-t-fit.minusi.t.llr
res.minusi.t.llr

H.t.llr<-H.lpr.fn(X.design,t,bopt_t,1)
H.t.llr

#Get variance weights for means model
```

```
varhat.llr<-exp(fit.t.llr)-1
varhat.llr
wts.t.llr<-1/varhat.llr
wts.t.llr
W.t.llr<-diag(wts.t.llr)

#Weigthted LLR FOR MEAN#
#Find optimal bandwidth from candidate list based on coded x's
par(mfrow=c(1,1))
band<-seq(.1,1,.01)
band.press_star_star.plot.w.fn(X.design,ybar,band,1,wts.t.llr)
band<-seq(.3,1,.01)
bopt_mean<-band.press_star_star.w.fn(X.design,ybar,band,1,wts.t.llr)$bopt
bopt_mean

#Find weighted LPR Estimates for mean
fit.mean.llr.w<-fit.lpr.w.fn(X.design,ybar,bopt_mean,1,W.t.llr)
fit.mean.llr.w
res.mean.llr.w<-ybar-fit.mean.llr.w
res.mean.llr.w

#Weighted Leave one out Cross-validation
fit.minusi.w.mean.llr<-fit.lpr.w.minusi.fn(X.design,ybar,bopt_mean,1,wts.t.llr)
fit.minusi.w.mean.llr
res.minusi.w.mean.llr<-ybar-fit.minusi.w.mean.llr
res.minusi.w.mean.llr

H.mean.llr.w<-H.lpr.w.fn(X.design,ybar,bopt_mean,1,W.t.llr)
H.mean.llr.w
#OPTIMIZE WITH GA#
#Mean target is 500
ga.llr.fn(X.design,ybar,500,bopt_mean,1,t,bopt_t,1,W.t.llr)

#SEMI-PARAMETRIC APPROACH#

#The following code will estimate responses semi-parametrically using MRR1 for the transformed variance and MRR2 for the mean#

#Preliminary Functions#
lambda.mrr1.fn<-function(y,yhat.p,yhat.np,yhat.minusi.p,yhat.minusi.np){
        #This will find asymptotically optimal data driven lambda for MRR1
        diff.minusi<-yhat.minusi.np-yhat.minusi.p
        res.p<-y-yhat.p
        diff<-yhat.np-yhat.p
        numerator<-t(diff.minusi)%*%res.p
        denominator<-t(diff)%*%diff
        lambda.mrr1<--(numerator/denominator)
        return(lambda.mrr1)
```

```
}
lambda.mrr2.fn<-function(y,yhat.p,rhat.llr){
        #This will find asymptotically optimal data driven lambda for MRR2
        res.p<-y-yhat.p
        numerator<-t(rhat.llr)%*%res.p
        denominator<-t(rhat.llr)%*%rhat.llr
        lambda.mrr2<-numerator/denominator
        opt<-min(lambda.mrr2,1)
        return(opt)
}
obj.dmrr.fn<-function(X,y,beta.mean,theta,b1,d1,l1,t,beta.t,b2,d2,l2,x0){
        #This function will find the estimated MSE=(yhat-theta)^2+varhat
#Given the data (X), the mean response (y), the coeffiecents for means wls model (beta.mean)
#the target value for the mean (theta), the bandwidth for the mean residuals (b1), the
#degree of the lpr for the mean (d1), the lambda for mean (l1) the transformed variance (t),
        #the coefficients for variance ols model
        #the bandwidth for the estimation of the transformed variance (b2), the
        #degree of the lpr for the transformed variance (d2), the lambda for variance (l2),
        #and the location of estimation (x0)
        x1.x2<-x0[1]*x0[2]
        x1.x3<-x0[1]*x0[3]
        x2.x3<-x0[2]*x0[3]
        x1.x1<-x0[1]*x0[1]
        x2.x2<-x0[2]*x0[2]
        x3.x3<-x0[3]*x0[3]
        modelx0.mean<-c(1,x0,x1.x2,x1.x3,x2.x3,x1.x1,x2.x2,x3.x3)
        modelx0.t<-c(1,x0)
        yhat.p<-beta.mean%*%modelx0.mean
        that.p<-beta.t%*%modelx0.t

        res.yhat<-y-yhat.p
        reshat.llr<-lpr.fn(X,res.yhat,b1,d1,x0)
        that.llr<-lpr.fn(X,t,b2,d2,x0)

        yhat.mmrr<-yhat.p+l1*reshat.llr
        that.vmrr<-l2*that.llr+(1-l2)*that.p

        bias<-theta-yhat.mmrr
        varhat<-exp(that.vmrr)-1
        msehat<-bias^2+varhat
        msehat

}

ga.dmrr.fn<-function(X,y,beta.mean,theta,b1,d1,l1,t,beta.t,b2,d2,l2){
        #This function will perform a Simple Genetic Algorithm
        #For the objective function estimated MSE=(yhat-theta)^2+var
```

```
        #Given the data (X), the mean response (y), the coeffiecents for means wls model (beta.mean)
#the target value for the mean (theta), the bandwidth for the mean residuals (b1), the
#degree of the lpr for the mean (d1), the lambda for mean (l1) the transformed variance (t),
#the coefficients for variance ols model
#the bandwidth for the estimation of the transformed variance (b2), the
#degree of the lpr for the transformed variance (d2), the lambda for variance (l2)

        #Stopping Criteria
        maxiter<-10000                          #set maximum number of iterations
        satiter<-1000       #set number of iterations during which the best results keep saturating
        epsln<-1e-8                             #smallest gain worth recognizing

        #GA Parameters
        m<-50                                   #population size at each generation
        mutrate<-.2                             #mutation rate
        crossrate<-.9                           #crossover rate
        zerorate<-.2                            #zero rate
        extremerate<-.2                         #extreme rate
        keep<-2                     #keep top two from each generation to remain unchanged
        num.parents<-m-keep         #number of population that will be used for mating
        mate<-ceiling((num.parents)/2)  #number of matings

        #Create initial population and initialize 'best result' holders
        xrange<-matrix(c(0,1,0,1,0,1),2,3)
        k<-ncol(xrange)
        elite<-matrix(0,keep,k)                 #matrix for elite chromosomes to remain unchanged
        cross<-matrix(0,num.parents,k)  #matrix for crossover
        iter<-0                                 #generation (iteration) counter
        stopcode<-0
        inarow<-0 #number of iterations with function's value consecutively less than epsln
        bestfun<-1e30                           #essential positive infinity
        bestx<-matrix(0,1,k)
        f<-rep(0,m)                             #initialize function value for each of m vectors
        G<-matrix(runif(m*k,0,1),m,k)   #initial generation with population size m

        #####Iterate through generations#####
        while(stopcode==0){
                iter<-iter+1                    #increments counter
if(iter>maxiter) stopcode<-2    #loop will exit on stopcode=2 for exceeding the maximum number of iterations

                #Evaluate current generation
                for(i in 1:m){
                        f[i]<-obj.dmrr.fn(X,y,beta.mean,theta,b1,d1,l1,t,beta.t,b2,d2,l2,G[i,])
                        }

        mat<-cbind(G,f)         #create matrix of x locations and corresponding function values
                newmat<-mat[order(mat[,4]),]    #sort matrix of by function values increasing
```

```
minf<-newmat[1,4]                    #optimal function (minimum)
bf<-min(minf, bestfun)
fgain<-bestfun-bf       #fgain is always non-negative it measure the change
if(fgain>epsln)         inarow<-0 else inarow<-(inarow+1)

if(fgain>0){
        bestfun<-bf
        bestx<-newmat[1,1:3]            #optimal x location
        }

if(inarow>satiter) stopcode<-1#loop will exit on stopcode=1 for best result having been achieved

#Select elite to remain unchanged
elite<-newmat[1:keep,1:3]

#Select parents for mating
cross<-newmat[(keep+1):m,1:3]

#Do Crossover
for(i in 1:mate){
        z<-rbinom(1,1,crossrate)
        if(z==1){
                x<-ceiling(runif(1,0,(k-1)))
                temp<-cross[i,(x+1):k]
                cross[i,(x+1):k]<-cross[i+mate,(x+1):k]
                cross[i+mate,(x+1):k]<-temp
                }
        }

#Do Mutation
M<-matrix(runif((num.parents*k),0,1),num.parents,k)
for(i in 1:num.parents){
        for(j in 1:k){
                zz<-rbinom(1,1,mutrate)
                if(zz==1) cross[i,j]<-M[i,j]
                }
        }
#Do Zero Gene Operator
for(i in 1:num.parents){
        for(j in 1:k){
                zzz<-rbinom(1,1,zerorate)
                if(zzz==1) cross[i,j]<-0
                }
        }
#Do Extreme Values Operator
Extreme<-matrix(1,num.parents,k)
for(i in 1:num.parents){
```

```
                        for(j in 1:k){
                                zzzz<-rbinom(1,1,extremerate)
                                if(zzzz==1) cross[i,j]<-Extreme[i,j]
                                }
                        }
                G<-rbind(elite,cross)
        }

        if(iter<maxiter) status<-0
        else status<-1

        result<-c(bestx,bestfun,iter)
        return(result)
}

#MRR1 FOR TRANSFORMED VARIANCE#

#Find optimal mixing parameter
lambda.opt_t<-c(lambda.mrr1.fn(t,fit.t.ols,fit.t.llr,fit.minusi.t.ols,fit.minusi.t.llr))
lambda.opt_t

#Find VMRR Estimates
fit.t.vmrr<-lambda.opt_t*fit.t.llr+(1-lambda.opt_t)*fit.t.ols
fit.t.vmrr
res.t.vmrr<-t-fit.t.vmrr
res.t.vmrr

#Find H_vmrr
H.t.vmrr<-lambda.opt_t*H.t.llr+(1-lambda.opt_t)*H.t.ols
H.t.vmrr

#Get variance weights for MMRR
varhat.t.vmrr<-exp(fit.t.vmrr) - 1
varhat.t.vmrr
wts.t.vmrr<-1/varhat.t.vmrr
wts.t.vmrr
W.t.vmrr<-diag(wts.t.vmrr)

#MRR2 FOR MEAN#
#LLR FOR Residuals#
#Find optimal bandwidth from candidate list based on coded x's
par(mfrow=c(1,1))
band<-seq(.1,1,.01)
band.press_star_star.plot.fn(X.design,res.mean.ewls,band,1)
band<-seq(.3,1,.01)
bopt_res<-band.press_star_star.fn(X.design,res.mean.ewls,band,1)$bopt
bopt_res
```

```
#Find LLR Estimates for residuals
fit.res.llr<-fit.lpr.fn(X.design,res.mean.ewls,bopt_res,1)
fit.res.llr
res.res.llr<-res.mean.ewls-fit.res.llr
res.res.llr

#Find H_res
H.res.llr<-H.lpr.fn(X.design,res.mean.ewls,bopt_res,1)
H.res.llr

#Semi-parametric Fits Using Means Model Robust Regression 2#
#Find optimal mixing parameter
lambda.opt_mean<-c(lambda.mrr2.fn(ybar,fit.mean.ewls,fit.res.llr))
lambda.opt_mean

#Find MMRR Estimates
fit.mean.mmrr<-fit.mean.ewls+lambda.opt_mean*fit.res.llr
fit.mean.mmrr
res.mean.mmrr<-ybar-fit.mean.mmrr
res.mean.mmrr

#Find H mmrr
I<-diag(1,27,27)
H.mean.mmrr<-H.mean.ewls+lambda.opt_mean*H.res.llr%*%(I-H.mean.ewls)
H.mean.mmrr

#OPTIMIZE WITH GA#
#Mean target is 500
ga.dmrr.fn(X.design,ybar,betas.mean.ewls,500,bopt_res,1,lambda.opt_mean,t,betas.t.ols,bopt_t,1,lambda.opt_t)
```