T-OPTIMAL DESIGNS FOR MODEL DISCRIMINATION IN PROBIT MODELS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Yue Ming

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Statistics

May 2014

Fargo, North Dakota

North Dakota State University

Graduate School

Title

T-OPTIMAL DESIGNS FOR MODEL DISCRIMINATION IN PROBIT MODELS

By

Yue Ming

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State university's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE

Dr. Seung Won Hyun

Dr. Gang Shen

Dr. Zhaohui Liu

Approved

05/05/2014	Dr. Rhonda Magel
Date	Department Chair

ABSTRACT

When dose-response functions have a downturn, one interesting feature to study is the significance of the downturn. The interesting feature can be studied using model discrimination between two rival models (model describing dose-response functions with a downturn versus model describing only increasing part of the response functions). In this article, we study T-optimal designs that can best discriminate between these two rival models. Three different sets of model parameter values are considered to demonstrate various shapes of dose-response functions. Under the different sets of the parameter values, the T-optimal designs are obtained, and their performances are compared to two other known designs for the model discrimination (D_s-optimal design and Uniform design) through Monte Carlo Simulation.

Keywords: Experimental design; T-Optimal Designs; Model Discrimination; Toxicology; Does-response

ACKNOWLEDGEMENT

I would like to thank my advisor, Dr. Seung Won Hyun, for his mentoring and support over the past year. I would also like to thank Dr. Gang Shen and Dr. Zhaohui Liu, for being my committee members and their expert guidance throughout my dissertation research. Many people at North Dakota State University assisted and encouraged me in various ways during my course of studies. My special thanks go to Dr. Rhonda Magel for her comments and suggestions on my dissertation.

I am grateful to my parents for giving me the spirit of excellence. I am thankful to my relatives and my friends for their encouragement and patience over the years. Without them, I could never have come so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In toxicological and biopharmaceutical research, dose-response experiments are widely conducted and nonmonotone dose-response curves (usually bell-shaped) are commonly observed. Experimenter hopes to gain assurance that a positive result reflects the phenomenon of interest by exposing the experimental units to a range of doses levels. During the experiments the chemical agent can have more than one effect at times, and hence nonmonotone dose-response relationships can occur. For instance, cellular toxicity at high doses can reduce the population of microbes on the plate and lower the frequency of visible colonies (Margolin, Kaplan and Zeiger, 1981).

For such a case, researchers are often interested in whether the downturn at higher doses is significant or not. From a statistical perspective, the study of the significance of this downturn can be transformed into discrimination between two rival models: a model describing dose-response functions with a downturn versus a model describing the increasing part of the response functions only. Welshons et al. (2003) considered a uniform design for studying MCF-7 human breast cancer cells vs. hormone concentration levels over the space $[-14, -4]$, and they found there is an up-and-down pattern. As we use Welshons' data as our motivating data, a probit regression model is adopted as it can provide sufficient descriptive information to non-linear dose-response functions, and it can describe the downturn at high dosages by adding a quadratic term in dose(Ting, 2006), thus provide better fit to the motivating data. Three sets of parameter values for η_1 are introduced concerning three different shapes of the dose-response function: Strong-downturn, Slight-downturn, and No-downturn.

Under the three dose-response function scenarios, we study T-optimal designs for discrimination between the two rival probit models with observations that are corrupted by normally distributed noise with mean zero and unknown variance. T-optimality criterion, introduced by Atkinson and Fedorov (1975a, b), maximizes the minimum distance between the two competing models, thus provide the most powerful F-test for lack of fit of the second model when the first is true. Since its introduction, numerous authors, such as Uciński and Bogacka (2005), Wiens (2009), Tommasi and López-Fidalgo (2010), etc., have studied the problem of obtaining T-optimal designs. The applications of T-optimal design have been found in many

important fields, see Atkinson, Bogacka and Bogacki (1998), Asprey and Macchietto (2000), Uciński and Bogacka (2005), Foo and Duffull (2011), etc..

Besides T-optimal design, Stigler (1971) and Studden (1982) in their early work determined optimal designs, namely D_s-optimal design, for discriminating between two nested univariate polynomial models. D_s-optimal design minimizes the volume of the confidence ellipsoid for the parameters corresponding to the extension of the smaller model, which refers to a likelihood ratio test. Liu (2013) studied and obtained D_s-optimal design for discriminating between the two rival probit models in her previous work based on Welshons' data (2003). We obtain the T-optimal designs, and we check its performance in the model discrimination against the D_s-optimal design obtained by Liu and a traditional uniform design used in Welshons et al. (2003) through Monte Carlo Simulation for each of the three dose-response function scenarios respectively.

In Chapter2, a brief introduction to the background of a dose-response study in toxicological study and optimal design theory is presented. Basic model set-up, and T-optimal designs under the probit models are presented in Chapter3. Discussion on the Monte Carlo Simulation procedure and performance comparisons are given in Chapter4. This paper is closing with discussions on future research topics in Chapter5.

CHAPTER 2. BACKGROUND

2.1. Dose-response

The regression of response on stimulus that is in the form of a dose (of a drug) can be represented as a curve, called a dose-response curve (Kotz and Johnson, 1982). In most cases, the dose-response model use the logarithm of dose (which is called the dose metameter), as we convert the X-axis values (doses) from arithmetic to logarithmic scale, the plotted dose-response curve will be changed from hyperbolic to sigmoid (”S” shape, stretched with steepest portion in the middle) and usually, between 25% to 75% of the Maximum response, the relation between doses and responses will be linear and therefore, better understanding and interpretation can be drawn from this linear area.

2.2. *T*-optimal designs

In toxicological studies, the mechanism of complex chemical reactions is not always fully known and experiments are done for their better understanding. Models of the processes can often be obtained from physical reasoning (pre-study, Phase I study, etc.), but there may be several plausible models. An experiment especially designed for discrimination between the competing models is a good source of information about the model fit using minimum experimental effort.

Atkinson and Fedorov (1975) consider designs for discrimination between two rival regression models $\eta_1(x, \Theta_1)$ and $\eta_2(x, \Theta_2)$. The models may be linear or nonlinear in the parameters which are estimated by least squares. Assume that the first model η_1 is true, so that the observations

$$y_{ij} = \eta_1(x_i, \Theta_1) + \epsilon_{ij}, \quad \epsilon_{ij} \sim N(0, \sigma^2), \tag{1}$$

(Note that $j = 1, 2, 3, \ldots, n_i$, where n_i is the number of subjects allocated to x_i, and the sample size $N = \sum_{i=1}^{k} n_i$. We will explain the model setup in detail in Chapter3.)

For a design ξ that puts weights w_i at k support points $x_i \in \mathcal{X}$, the lack of fit sum of squares for model $\eta_2(x, \Theta_2)$ is made as large as possible by maximizing

$$\triangle(\xi) = \int_{i=1}^{k} (\eta_1(x_i, \Theta_1) - \eta_2(x_i, \hat{\Theta}_2))^2 dx, \tag{2}$$

where $\hat{\Theta}_2$ is the estimated parameters of η_2 that minimize its distance from η_1(*see Section 3.2*).

The design maximizing (*2*) is called *T*-optimal design, denoted as ξ_T^*. Under some regularity conditions Atkinson and Fedorov (1975) proved the following Equivalence Theorem about *T*-optimal designs.

THEOREM 1.

(i) a design ξ_T^* is *T*-optimal if and only if

$$\psi(x, \xi_T^*) \leqslant \triangle(\xi) \quad x_i \in \mathcal{X} \tag{3}$$

where

$$\psi(x, \xi) = (\eta_1(x, \Theta_1) - \eta_2(x, \hat{\Theta}_2))^2 \tag{4}$$

(ii) the upper bound of $\psi(x, \xi_T^*)$ is achieved at the points of the *T*-optimal design;

(iii) for any non-optimal design ξ, for which $\triangle(\xi) < \triangle(\xi_T^*)$,

$$\sup_{x \in \mathcal{X}} \psi(x, \xi) > \triangle(\xi_T^*). \tag{5}$$

These conditions include the continuity and differentiability with respect to the parameters of the models.

Nested models are considered in our case. In general, the parameter estimate of the false model $\hat{\Theta}_2$, as (*2*) and (*4*) show, will depend on the design ξ and on Θ_1. Thus *T*-optimal designs are often locally optimum.

2.3. D_s**-optimal designs**

If the main interest is to test whether the expected mean responses of the nested models are the same, the problem would lead to test whether the subset of parameters equal to 0. D_s-optimal design maximizes the determinant of the Fisher Information Matrix for estimating the sub-parameters accurately, thus the power for testing if the sub-parameters equal to 0 is maximized.

CHAPTER 3. *T*-OPTIMAL DESIGNS FOR MODEL DISCRIMINATION

3.1. Models

Recall the the general non-linear regression model in Chapter2, we assume that η is a real-valued continuous function of both arguments $(x, \theta) \in \mathcal{X} \times \Theta$ and a design is defined as a probability measure ξ on $\mathcal{X}$ with finite support (see Kiefer (1974)). If the design ξ has masses w_i at point $x_i \ \ (i = 1, \ldots, k)$ and N observations can be made by the experimenter, then the quantities $w_i n$ are rounded to the nearest integers, say n_i, that satisfying $N = \sum_{i=1}^{k} n_i$, and n_i observations are taken at each location $x_i \ \ (i = 1, \ldots, k)$.

We assume the model $\eta_1(x, \Theta_1)$ (with a quadratic term) is an extension of model $\eta_2(x, \Theta_2)$ (without quadratic term) and can be defined in the form below,

$$\eta_1(x, \Theta_1) = \Phi(-(\theta_{11} + \theta_{12} \cdot x_i + \theta_{13} \cdot x_i^2) \tag{6}$$

$$\eta_2(x, \Theta_2) = \Phi(-(\theta_{21} + \theta_{22} \cdot x_i) \tag{7}$$

where Φ is a cumulative standard normal distribution (a probit model).

We are interested in finding the design that can best discriminate between these two competing (nested) models, in other words, the best design that maximizes the power for testing the hypothesis

$$H_0 : \eta = \eta_1 \quad vs. \quad H_a : \eta = \eta_2 \tag{8}$$

which corresponds in the context of nested models to the hypothesis

$$H_0 : \theta_{13} = 0 \quad vs. \quad H_a : \theta_{13} \neq 0 \tag{9}$$

As we mentioned, Welshons et al. (2003) considered a uniform design for studying dose-response function with a downturn over the hormone concentration range (design space) of

$\mathcal{X} = [-14, -4]$, and Liu (2013) studied to search D_s-optimal design for the same response function over the same design space. Therefore, in order to demonstrate the dose-response functions in this study, as well as making the comparison among the three designs valid and comprehensive, we adopt the same parameter settings as follows,

Table 1: Three sets of parameters for η_1

Case	θ_1	θ_2	θ_3
$\Theta_{1,1}$	4.63	1.23	0.07
$\Theta_{1,2}$	0.175	0.277	0.024
$\Theta_{1,3}$	-6.69	-0.60	0.01

Each parameter-set describes a specific dose-response function scenario: (a) Strong downturn; (b) Slight downturn; (c) No downturn (*see Figure 1*). We obtain T-optimal design under these three scenarios.

3.2. Method: *T*-optimality

T-Optimality criterion determines the design ξ that maximizes the minimum deviation between the full model η_1 and reduced model η_2, that is,

$$\xi = arg \ \max_{x_i \in \mathcal{X}} \sum_{i=1}^{k} w_i \cdot (\eta_1(x_i, \Theta_1) - \eta_2(x_i, \hat{\Theta}_2))^2 \tag{10}$$

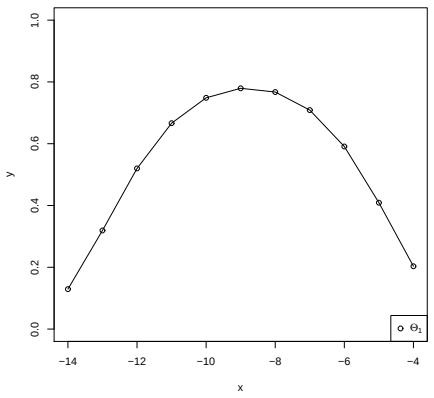
where the parameters $\hat{\Theta}_2$ of model η_2 can be obtained by numerical search method when it gives the best approximation to the full model η_1, which is determined as

$$\hat{\Theta}_2 = arg \ \min_{\Theta_2 \in \Theta_S} \sum_{i=1}^{k} w_i \cdot (\eta_1(x_i, \Theta_1) - \eta_2(x_i, \hat{\Theta}_2))^2 \tag{11}$$

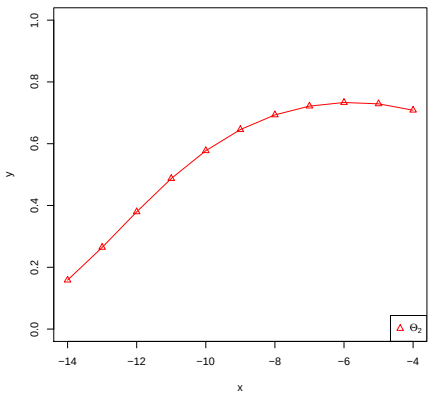
where Θ_S is the estimated range for Θ_2, from which we search for the best estimates.

Thus the T-optimal design $\psi(x, \xi)$ can be obtained if and only if the following optimality equivalence condition (Kiefer 1974) is met at the obtained points of the optimal design,

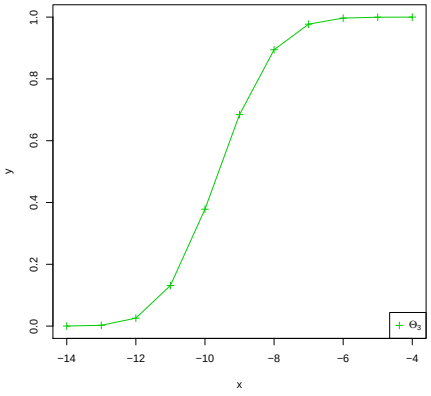
$$(\eta_1(x, \Theta_1) - \eta_2(x, \hat{\Theta}_2))^2 - \sum_{i=1}^{k} w_i \cdot (\eta_1(x_i, \Theta_1) - \eta_2(x_i, \hat{\Theta}_2))^2 \leqslant 0 \tag{12}$$



(a) Strong Downturn



(b) Slight Downturn



(c) No Downturn

Figure 1: Dose-response curve of the three scenarios

We use numerical method to find T-optimal design, and the algorithm can be realized by the following iterative procedure:

- *Step 1:* For any given design ξ_s (initial design) supported at points $x_1, x_2, \ldots, x_k$, take

$$\hat{\Theta}_{2,s} = arg \min_{\Theta_{2,s} \in \Theta_S} \sum_{i=1}^{k} w_{i,s} \cdot (\eta_1(x_{i,s}, \Theta_1) - \eta_2(x_{i,s}, \hat{\Theta}_{2,s}))^2.$$

- *Step 2:* Find the point $x_{s+1} = arg \ max_{x_{s+1} \in \mathcal{X}} \ (\eta_1(x_{s+1}, \Theta_1) - \eta_2(x_{s+1}, \hat{\Theta}_{2,s}))^2$.

- *Step 3:* $s = s + 1$.

- *Step 4:* Assign weight $w_s = \frac{1}{s+1}$ to the new point x_s, and we have

$$\xi_{new} = \left\{ \begin{array}{c} x_s \\ w_s \end{array} \right\}.$$

- *Step 5:* Construct the new design $\xi_s = (1 - w_s) \cdot \xi_{s-1} + w_s \cdot \xi_{new}$.
- *Step 6:* Check *T*-optimality equivalence condition (*12*).

The procedure will stop when *T*-optimality equivalence condition (*12*) is achieved. Thus ξ_s is the *T*-optimal design we are looking for.

3.3. Obtaining *T*-optimal designs

To begin the iterative procedure, we use the traditional uniform design over the previously mentioned design space $\mathcal{X} = [-14, -4]$ as the initial design, that is,

$$\xi_0 = \left\{ \begin{array}{cccc} -14 & -10 & -6 & -4 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{array} \right\} \tag{13}$$

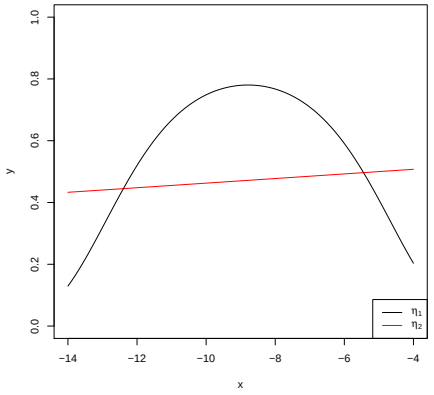
For each of the three cases, We ran the iteration respectively and we obtained the estimated parameters $\hat{\Theta}_2$ for model η_2, which are given in *Table 2* below,

Table 2: Estimated model parameters for η_2

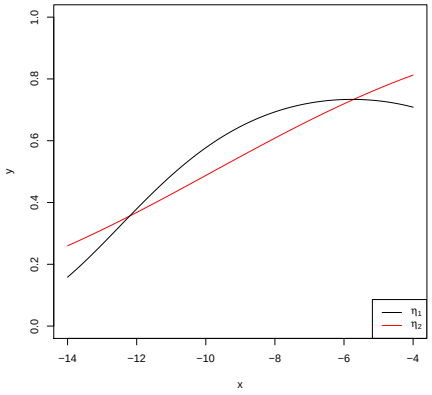
Case	η_1			η_2	
	θ_1	θ_2	θ_3	$\hat{\theta}_1$	$\hat{\theta}_2$
Θ_1	4.63	1.23	0.07	-0.0938	-0.0188
Θ_2	0.175	0.277	0.024	-1.484	-0.153
Θ_3	-6.69	-0.60	0.01	-5.734	-0.598

As the theory says, $\eta_2(x, \hat{\Theta}_2)$ should be the best approximation of $\eta_1(x, \Theta_1)$, and it is represented by the curve in red (respectively) in *Figure 2*, while the full model η_1 is represented by the curve in black.

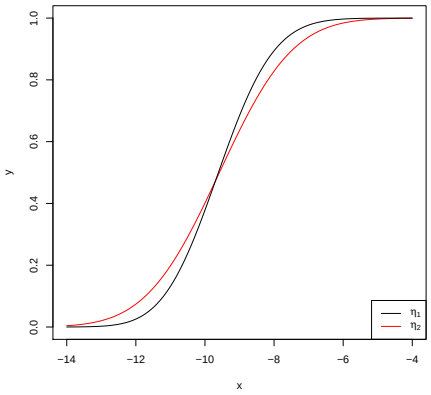
Hence, we obtained the *T*-optimal design that satisfies (*10*) under the estimated $\eta_2(x, \hat{\Theta}_2)$ for the three cases, say $\xi^*_{T_1}$, $\xi^*_{T_2}$, $\xi^*_{T_3}$, and we list them as follows,



(a) Θ_1



(b) Θ_2



(c) Θ_3

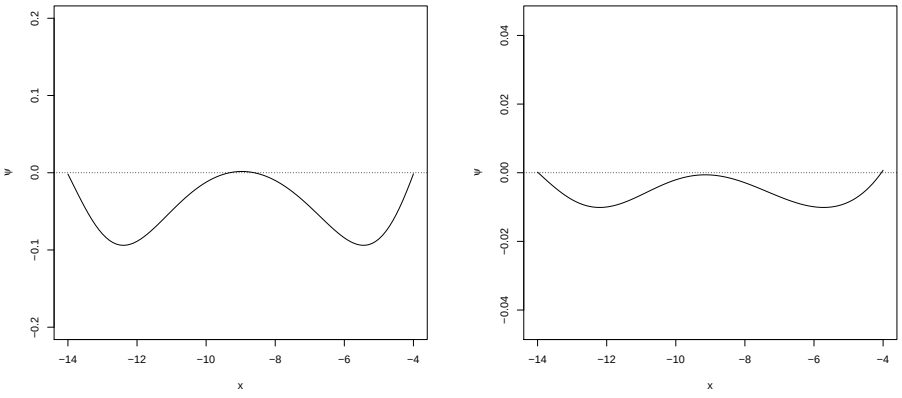
Figure 2: Plots of the two rival models

$$\xi^*_{T_1} = \left\{ \begin{array}{ccc} -14 & -9 & -4 \\ 0.251 & 0.498 & 0.249 \end{array} \right\} \tag{14}$$

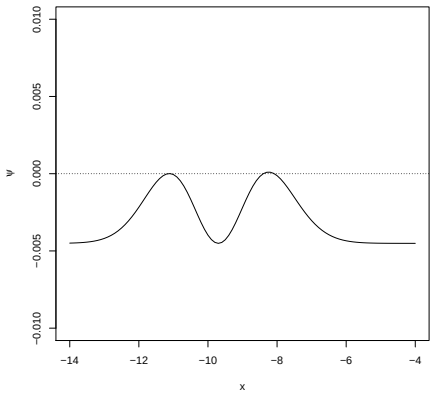
$$\xi^*_{T_2} = \left\{ \begin{array}{ccc} -14 & -9.17 & -4 \\ 0.275 & 0.425 & 0.299 \end{array} \right\} \tag{15}$$

$$\xi^*_{T_3} = \left\{ \begin{array}{cc} -11.1 & -8.2 \\ 0.392 & 0.557 \end{array} \right\} \tag{16}$$

In order to verify that the obtained designs are really locally optimal, we calculate sensitive function (*12*) over $\mathcal{X} = [-14, -4]$, and the results show that at each of the obtained design points, the sensitive function reaches the locally maximum value of 0 (with tolerance level<0.0001) (*see Figure 3*), thus it confirms that each of the obtained designs is a T-optimal design for a given set of parameters.



(a) Θ_1 (b) Θ_2



(c) Θ_3

Figure 3: Optimality Verification of the locally T-optimal design

CHAPTER 4. PERFORMANCE

4.1. Efficiency

Design efficiency is a measure which compares a given design to the optimal design. Here the efficiency of a design ξ is $H(\xi)$, then the design would need $\lambda = 100 \times (\frac{1}{H(\xi)} - 1)\%$ more subjects than the *T*-optimal design to achieve the same level of accuracy for the model discrimination. The efficiency of a design ξ is computed as,

$$H(\xi) = \frac{\sum_j w_j \cdot (\eta_1(x_j, \Theta) - \eta_2(x_j, \hat{\Theta}))^2 | \xi}{\sum_i w_i \cdot (\eta_1(x_i, \Theta) - \eta_2(x_i, \hat{\Theta}))^2 | \xi_T^*} \tag{17}$$

Recall *Equation 10*, $\hat{\Theta}$ minimizes the distance between the two models under the specified design. Because *T*-optimal design maximizes this minimum distance, we want to compare *T*-optimal design to D_s- and uniform designs. If D_s- or uniform design works close to *T*-optimal design for model discrimination, $H(\xi)$ will be close to 1, otherwise $H(\xi)$ will be far from 1.

Table 3: Competing designs for the three sets of parameters for η_1

Case	ξ_T^*	$\xi_{D_s}^*$	ξ_U
Θ_1	$\begin{Bmatrix} -14 & -9 & -4 \\ 0.251 & 0.498 & 0.249 \end{Bmatrix}$	$\begin{Bmatrix} -13.84 & -8.84 & -4 \\ 0.285 & 0.467 & 0.248 \end{Bmatrix}$	$\begin{Bmatrix} -14 & \dots & -4 \\ \frac{1}{11} & \dots & \frac{1}{11} \end{Bmatrix}$
Θ_2	$\begin{Bmatrix} -14 & -9.17 & -4 \\ 0.275 & 0.425 & 0.299 \end{Bmatrix}$	$\begin{Bmatrix} -14 & -9.06 & -4 \\ 0.337 & 0.431 & 0.232 \end{Bmatrix}$	$\begin{Bmatrix} -14 & \dots & -4 \\ \frac{1}{11} & \dots & \frac{1}{11} \end{Bmatrix}$
Θ_3	$\begin{Bmatrix} -11.1 & -8.2 \\ 0.418 & 0.582 \end{Bmatrix}$	$\begin{Bmatrix} -11.54 & -9.57 & -7.49 \\ 0.381 & 0.217 & 0.402 \end{Bmatrix}$	$\begin{Bmatrix} -14 & \dots & -4 \\ \frac{1}{11} & \dots & \frac{1}{11} \end{Bmatrix}$

Table 3 listed the three designs to be compared. Noted that ξ_T^* for case Θ_3 has only two design points. Since *T*-optimal design only consider to maximize the minimum distance between the rival models, it is possible to have the number of *T*-optimal design points less than the number of model parameters. If the number of design points is less than the number of model parameters, the parameters are not estimable. In order to void this problem, we add a third point $x_i \in \mathcal{X}$ that is different from both -11.1 and -8.2 with a small weight to $\xi_{T_3}^*$,

$$\xi_{T_3 Sub}^* = \begin{Bmatrix} -11.1 & \textbf{-10} & -8.2 \\ 0.393 & \textbf{0.050} & 0.557 \end{Bmatrix} \tag{18}$$

(Note that the total weight adds up to 1.)

Here the efficiency of the substitute design $H(\xi^*_{T_3Sub}) = 0.9684$, which represents that the sub design will need only 3.26% more subjects to achieve the same accuracy level as the original *T*-optimal design $\xi^*_{T_3}$ does, thus it has provided sufficient evidence that this modified design is feasible as a substitution.

Table 4: Performance for the model discriminations

Case	ξ^*_T		$\xi^*_{D_s}$		ξ_U	
	H	λ	*H*	λ	*H*	λ
Θ_1	1	(0%)	0.955	(4.8%)	0.560	(78.7%)
Θ_2	1	(0%)	0.996	(0.4%)	0.553	(80.8%)
Θ_3	1	(0%)	0.611	(63.6%)	0.334	(199.9%)

The efficiency comparisons between *T*-, D_s- and uniform designs are given in *Table 4*. It shows that D_s-optimal designs perform as well as *T*-optimal designs for both strong-downturn and slight-downturn cases, but a fairly poor efficiency of 0.611 for the third case Θ_3. It indicates that at least 63.6% more subjects will be needed to provide the same level of accuracy as *T*-optimal design for the model discrimination when the model has no-downturn effect at high doses. Uniform designs perform poorly overall performance for all three scenarios of the dose-response functions, especially for the third case when no-downturn occurs, the number of subjects needed to reach the same level of accuracy are almost three times larger than the modified *T*-optimal design $\xi^*_{T_3Sub}$.

4.2. Simulation

In this section, we implement Monte Carlo Simulation to check the performance of the three designs for the model discrimination.

Recall *Equation 1*, the only factor that makes the observation different from each other is the random error. Therefore, under each of the three probit models, we simulate a sample of 100 observations with normally distributed random error of mean 0 and $\sigma^2 = 0.01$, and run the likelihood-ratio test (with $\alpha = .05$) for 1000 times testing $H_0 : \eta = \eta_1 \quad vs. \quad H_a : \eta = \eta_2$ under *T*-optimal design, D_s-optimal design and Uniform design separately. We compute both the power and the significance level of the test, and the result is shown below,

Table 5: Test power and α level of the competing designs for model discrimination

Case	ξ_T^*		$\xi_{D_s}^*$		ξ_U	
	Power	α	*Power*	α	*Power*	α
Θ_1	0.915	0.053	0.684	0.065	0.597	0.050
Θ_2	0.997	0.049	0.798	0.051	0.505	0.046
Θ_3	0.983	0.045	0.836	0.048	0.684	0.066

From *Table 5*, we can see that for each of the designs, the computed α-levels are all close to 0.05, which proves that all the designs in our research maintain an appropriate level of Type-I-Error. The power of the *F*-test shows that *T*-optimal designs provide significantly better performances than D_s-optimal designs and uniform designs under all three dose-response scenarios.

CHAPTER 5. CONCLUSION

We studied T-optimal design and its performance for model discrimination in probit models compared to other designs. By introducing three sets of parameter values to the full model η_1, we have dose-response function subjected to three scenarios: Strong-downturn, Slight-downturn and No-downturn. Under each of the case, we applied a numerical method to search the T-optimal design that maximizes the distance between the full model η_1 and the estimated reduced model η_2 that gives the best approximation to η_1.

We compared the efficiency of the D_s-optimal designs (obtained by Liu, 2013) and traditional uniform designs with T-optimal design and the results indicate that:

- D_s-optimal design works as well as T-optimal design when the dose-response function is subjected to a Strong-downturn or Slight-downturn effect; but a fairly poor performance when No-downturn effect is expected.
- Uniform design works poorly for all three dose-response scenarios.

We then performed Monte Carlo Simulation to compute the power of the likelihood-ratio test ($\alpha = 0.05$) for the model discrimination provided by each of the three designs under the three scenarios separately. Results show that:

- Each simulated significance level α is very close to 0.05 that was used in the LR test, which indicates that all three designs have maintained an appropriate level of Type-I-Error.
- T-optimal design provides the most powerful test for discriminating between the two probit models under all three dose-response scenarios, while D_s-optimal design provides a less powerful test for model discrimination. Uniform design provides the lowest test power for all three cases.

One of the fundamental assumptions in the optimal design theory for non-linear model is that the response function and the parameter values must be known. As we were using numerical method for searching T-optimal design, we assume that we know η_1, and we have

assigned fixed parameter values to the full model η_1. However, this is not always practical in a real world problem.

To avoid being dependent on the constraints of parameter values, numerous authors, such as Hunter $\&$. Reiner (1965), Pazman $\&$. Fedorov (1968) and Fedorov (1971), have studied sequential experimental designs for model discrimination, which are less dependent on the constraints on the values of the parameters, known as the asymptotically optimality, which will be included in our future work as an extension.

REFERENCES

[1] Kotz, S. and Johnson N. L. (1982). *Encyclopedia of Statistical Sciences uol. 2*, Wiley, New York, p. 418.

[2] Margolin, B. H., Kaplan N. and Zeiger, E. (1981). *Statistical analysis of the Ames Salmonella/microsome test*

[3] Atkinson, A. C. and Fedorov, V. V. (1975a). *The design of experiments for discriminating between two rival models*, *Biometrika* **62** 57C70.

[4] Atkinson, A. C. and Fedorov, V. V. (1975b). *Optimal design: Experiments for discriminating between several models*, *Biometrika* **62** 289C303.

[5] Uciński, D. and Bogacka, B. (2005). *Bayesian optimum designs for discriminating between models with any distribution*, Compute. Statist. Data Anal. **54** 143C150.

[6] Wiens, D. P. (2009). *Robust discrimination designs*, J. R. Stat. Soc. Ser. B Stat. Methodol. **71** 805-829.

[7] Tommasi, C. and López-Fidalgo, J. (2010). *T-optimum designs for discrimination between two multiresponse dynamic models*, J. R. Stat. Soc. Ser. B Stat. Methodol. **67** 3-18.

[8] Atkinson, A. C., Bogacka, B. and Bogachi, M. B. (1998). *D- and T-optimum designs for the kinetics of a reversible chemical reaction*, Chemometrics and Intelligent Laboratory Systems **43** 185-198.

[9] Asprey, S. P and Macchietto, S. (2000). *Statistical tools for optimum dynamic model building*, Computers and Chemical Engineering **24** 1261-1267.

[10] Foo, L. K. and Duffull, S. (2011). *Optimal design of pharmacokinetic-pharmacodynamic studies*, Pharmacokinetics in Drug Development, Advances and Applications, **3** 175-194.

[11] Hunter, W. G. and Reiner, A. M. (1965). *Designs for discriminating between two rival models*, Technometrics **7** 307-23

[12] Pazman, A. and Fedorov, V. V. (1968). *Planning of regression and discrimination experiments on NN scattering*, Sov. J. Nucl. Phys. **6** 619-21

[13] Fedorov, V. V. (1971). *Asymptotically optimal designs of experiments for discriminating two rival regression models*, Theory Prob. Applic. **16** 561-2

[14] Kiefer, J. (1974). *General equivalence theory for optimum designs*, Ann. Statist. **2** 849C879.

[15] Stigler, S. (1971). *Optimal experimental design for polynomial regression*, J. Amer. Statist. Assoc. **66** 311C318.

[16] Studden, W. J. (1982). *Some robust-type D-optimal designs in polynomial regression*, J. Amer. Statist. Assoc. **77** 916C921. MR0686418.

[17] Graybill, F.A. (1983). *Matrices with Applications in Statistics, 2nd Edition*, Wadsworth International Group, Belmont, CA.

[18] Ting, N. (2006). *Dose Finding in Drug Development*, New York.

APPENDIX A. R-CODE FOR OBTAINING *T*-OPTIMAL DESIGNS

```
#Estimated Model 2 parameters' range:
#theta1 [-10,0]
#theta2 [-0.01, -1]

#True model with Theta1:
#theta1 = 4.63     theta1 = 0.175     theta1 = -6.69
#theta2 = 1.23     theta2 = 0.277     theta2 = -0.60
#theta3 = 0.07     theta3 = 0.024     theta3 = 0.01

#Initial value#

x0 = c(-14, -10, -6, -4)
n0 = length(x0)
w = rep(1/n0, n0)
(D = rbind(x0, w))
p = 1
n = 1
while(p > .0001){
Rt1 = c(-5, 5)
Rt2 = c(-1, 1)
s = c(1, .1)
   while(max(s) > .01){
theta1 = seq(Rt1[1], Rt1[2], s[1])
theta2 = seq(Rt2[1], Rt2[2], s[2])

eta1 = function(x){
pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}

mod2 = expand.grid(theta1, theta2)

diff = rep(NA, nrow(mod2))
for (i in 1:nrow(mod2)){
diff[i] = sum(w*(sapply(x0, eta1) - pnorm(-(mod2[i,1]
        + mod2[i,2]*x0)))^2)
}
(theta1hat = mod2[which.min(diff),1]); (theta2hat = mod2[
        which.min(diff),2])
s = s/2
Rt1[1] = theta1hat - s[1]
Rt1[2] = theta1hat + s[1]
Rt2[1] = theta2hat - s[2]
Rt2[2] = theta2hat + s[2]
}
theta1hat
```

```
theta2hat
x = seq(-14, -4, .1)
a = rep(NA, length(x))
diff2 = rep(NA, length(x))
for (j in 1:length(x)){
diff2[j] = (sapply(x[j], eta1) - pnorm(-(theta1hat +
        theta2hat *x[j])))^2
}
(anew = x[which.max(diff2)])
p = abs((sapply(anew, eta1) - pnorm(-(theta1hat +
        theta2hat*anew)))^2 - sum(w*(sapply(x0, eta1) -
        pnorm(-(theta1hat + theta2hat*x0)))^2))
x0 = c(x0,anew)
alpha = 1/(n + 1)
w = c((1 - alpha) * w, alpha)
n = n + 1
print(p)
D = rbind(x0,w)
}

#Summarize the result

T_optimal = by(D[2,], D[1,], FUN = sum)
T_optimal
#plot the two models
theta1hat
theta2hat
y = pnorm(-(theta1hat + theta2hat*x))
y1 = eta1(x)
plot(x, y, cex = 0.3, ylim = c(0, 1), col = "blue",
        type="l",pch=1)
lines(x, y1,type="l",pch=1)
cont.txt=c(expression(paste(eta) [1]),
        expression(paste(eta) [2]))
legend("bottomright", legend=cont.txt,
        col=c(1,4), lwd=1, lty=c(1,1))

#Verify T-optimal

X = D[1,]
W = D[2,]
x = seq(-14, -4, .1)
ds = rep(0,length(x))
for (i in 1:length(x))
{ds[i] = (sapply(x[i], eta1) - pnorm(-(theta1hat +
        theta2hat*x[i])))^2 - sum(W*(sapply(X, eta1) -
        pnorm(-(theta1hat + theta2hat*X)))^2)}
cont.txt2 = (expression(psi))
```

```
plot(x, ds, cex = 0.3, type = "l", pch=1, ylab = cont.txt2,
        ylim = c(-0.09400517,0))
abline(h = 0,pch =1, lty = 3)
abline(v = c(-14, -9, -4), pch =1, lty = 3)
```

APPENDIX B. R-CODE FOR CALCULATING EFFICIENCY

```
#Obj function:
sum(w*(sapply(x0, eta1) - pnorm(-(theta1hat + theta2hat*x0)))^2)

#########
#THETA-1#
#########
theta1hat = -0.09375; theta2hat = -0.01875
eta1 = function(x){
pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
#T-optimal
x0 = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)

#Obj = 0.0940061

#Ds-optimal
x0 = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)

#Obj = 0.08972737
EffDs1 = 0.08972737/0.0940061 #0.9544845

#Uniform
x0 = seq(-14, -4, by = 1)
w = rep(1/length(x0), length(x0))

#Obj = 0.05260749
EffU1 = 0.05260749/0.0940061 #0.5596178

#########
#THETA-2#
#########
theta1hat = -1.484375; theta2hat = -0.153125
eta1 = function(x){
pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
#T-optimal
x0 = c(-14, -9.17, -4)
w = c(0.2749443, 0.4247492, 0.2989130)

#Obj = 0.01011802

#Ds-optimal
x0 = c(-14, -9.06, -4)
```

```
w = c(0.337, 0.431, 0.232)

#Obj = 0.0100757
EffDs2 = 0.0100757/0.01011802 #0.9958174

#Uniform
x0 = seq(-14, -4, by = 1)
w = rep(1/length(x0), length(x0))

#Obj = 0.005393798
EffU2 = 0.005393798/0.01011802 #0.5330883

#########
#THETA-3#
#########
theta1hat = -5.734375; theta2hat = -0.5984375
eta1 = function(x){
pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}

#T-optimal
x0 = c(-11.1, -8.2)
w = c(0.4176051, 0.5823947)

#Obj = 0.00433816

#Sub-design for T3
x0 = c(-11.1, -10, -8.2)
w = c(0.393, 0.050, 0.557)

#Obj = 0.004366865
EffT3sub = 0.004366865/0.004509453 #0.9683803

#Ds-optimal
x0 = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)

#Obj = 0.002667478
EffDs3 = 0.002667478/0.004366865 #0.61084508

#Uniform
x0 = seq(-14, -4, by = 1)
w = rep(1/length(x0), length(x0))

#Obj = 0.001455079
EffU3 = 0.001455079/0.004366865 #0.3334648
```

APPENDIX C. R-CODE FOR SIMULATION UNDER *T*-OPTIMAL DESIGNS

```
library(lmtest)
library(nls2)

N = 1000
mu = 0
sigma = sqrt(.01)


#########
#THETA-1#
##########################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##########################################


#################
#sample-size 25#
#################
n1 = 25

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
w = round(w, 2)
(ns = n1*w)
(ns = round(round(ns,1) + 0.0001, 0))


#################
#sample-size 50#
#################
n1 = 50

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 75#
```

```
################
n1 = 75

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 100#
################
n1 = 100

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}
```

```
Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 4.63 1.23 0.07
#fitted theta: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.09, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
```

```
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.09, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #.915


#########
#THETA-2#
############################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
############################################


#################
#sample-size 25#
#################
n1 = 25

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns + 0.02, 0))

#################
#sample-size 50#
#################
n1 = 50

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns + 0.06, 0))


################
```

```
#sample-size 75#
################
n1 = 75

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E =sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}
```

```
Error1ForPoint4 = matrix(NA, N, ns[4], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]), rep(y1[4], ns[4]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]), rep(x[4], ns[4]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .01)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
```

```
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
for(i in 1:1000){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm  ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #.997


#########
#THETA-3#
##########################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}
##########################################


################
#sample-size 25#
################
n1 = 25

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w + )
(ns = round(ns, 0))


################
#sample-size 50#
```

```
################
n1 = 50

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
```

```
}
}


Error1 = cbind(Error1ForPoint1, Error1ForPoint2)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
```

```
for(i in 1:1000){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) # 0.983


###########
###########
#  ALPHA  #
###########
###########

N = 1000
mu = 0
sigma = sqrt(.01)


#########
#THETA-1#
##############################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##############################################

#################
#sample-size 25#
#################
n1 = 25

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
w = round(w, 2)
(ns = n1*w)
(ns = round(round(ns,1) + 0.0001, 0))
```

```
################
#sample-size 50#
################
n1 = 50

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-14, -9, -4)
w = c(0.2512376, 0.4981998, 0.2489874)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
```

```
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta:4.63 1.23 0.07
#thetahat: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
```

```
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.3, beta2 = 1.2, beta3 = 0.07),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 = -0.01),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.053


#########
#THETA-2#
##########################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
##########################################

#################
#sample-size 25#
#################
n1 = 25

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns + 0.02, 0))
```

```
################
#sample-size 50#
################
n1 = 50

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns + 0.06, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-14, -9.2, -9.1, -4)
w = c(0.2749443, 0.2993311, 0.1254181, 0.2989130)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
```

```
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[4], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}


Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]), rep(y1[4], ns[4]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]), rep(x[4], ns[4]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
```

```
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}
W = na.omit(W)
(alpha = sum(W < 0.05)/length(W)) #.04888889


#########
#THETA-3#
##########################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}
##########################################
```

```
################
#sample-size 25#
################
n1 = 25

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 50#
################
n1 = 50

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-11.1, -8.2)
w = c(0.3924051, 0.556962)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######
```

```
Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}


Error1 = cbind(Error1ForPoint1, Error1ForPoint2)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.6, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
```

```
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.6, beta3 = 0.01),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.045
```

APPENDIX D. R-CODE FOR SIMULATION UNDER D_S-OPTIMAL DESIGNS

```
N = 1000
mu = 0
sigma = sqrt(.01)

#########
#THETA-1#
##########################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##########################################


#################
#sample-size 25#
#################
n1 = 25

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
(ns = round(ns, 0))

#################
#sample-size 50#
#################
n1 = 50

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w+0.11)
(ns = round(ns, 0))


#################
#sample-size 75#
#################
n1 = 75

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
```

```
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
```

```
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 4.63 1.23 0.07
#fitted theta: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.09, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.09, beta2 = -0.01),
```

```
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #.684


#########
#THETA-2#
##########################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
##########################################


#################
#sample-size 25#
#################
n1 = 25

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 50#
#################
n1 = 50

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns - 0.06, 0))


#################
#sample-size 75#
#################
n1 = 75

x = c(-14, -9.06, -4)
```

```
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns + 0.11, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E =sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}


Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1
```

```
y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .01)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
for(i in 1:1000){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
```

```
        algorithm  ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #0.798


#########
#THETA-3#
##########################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}
##########################################


#################
#sample-size 25#
#################
n1 = 25

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 50#
#################
n1 = 50

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 75#
#################
```

```
n1 = 75

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
```

```
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
for(i in 1:1000){
K1 = F1[i,]
```

```
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) # 0.836


###########
###########
#  ALPHA  #
###########
###########


N = 1000
mu = 0
sigma = sqrt(.01)


#########
#THETA-1#
##########################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##########################################

################
#sample-size 25#
################
n1 = 25

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
(ns = round(ns, 0))

################
#sample-size 50#
```

```
################
n1 = 50

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w+0.11)
(ns = round(ns, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-13.84, -8.84, -4)
w = c(0.285, 0.467, 0.248)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
```

```
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta:4.63 1.23 0.07
#thetahat: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2 = 1.2, beta3 = 0.06),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
```

```
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.3, beta2 = 1.2, beta3 = 0.07),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 = -0.01),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.065


#########
#THETA-2#
##########################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
##########################################


################
#sample-size 25#
################
n1 = 25

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns, 0))
```

```
################
#sample-size 50#
################
n1 = 50

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns - 0.06, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns + 0.11, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-14, -9.06, -4)
w = c(0.337, 0.431, 0.232)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
```

```
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}


Error1 = cbind(Error1ForPoint1, Error1ForPoint2, Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
```

```
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.051


#########
#THETA-3#
###############################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}
###############################################

#################
#sample-size 25#
#################
n1 = 25

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


################
```

```
#sample-size 50#
################
n1 = 50

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


################
#sample-size 75#
################
n1 = 75

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


#################
#sample-size 100#
#################
n1 = 100

x = c(-11.54, -9.57, -7.49)
w = c(0.381, 0.217, 0.402)
(ns = n1*w)
(ns = round(ns, 0))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
```

```
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma * rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]),
        rep(y1[3], ns[3]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]),
        rep(x[3], ns[3]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
```

```
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.048
```

APPENDIX E. R-CODE FOR SIMULATION UNDER UNIFORM DESIGNS

```
N = 1000
mu = 0
sigma = sqrt(.01)


#########
#THETA-1#
##########################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##########################################


#################
#sample-size 100#
#################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
```

```
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
```

```
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation
```

```
##################################
#Fit a model to simulated data 1#
#original theta: 4.63 1.23 0.07
#fitted theta: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2  = 1.2, beta3 = 0.07),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.1, beta2 = -0.02),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 =  4.6, beta2 = 1.2, beta3 = 0.07),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.09, beta2 =  -0.02),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #.597
```

```
#########
#THETA-2#
##########################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
##########################################


#################
#sample-size 100#
#################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
```

```
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
```

```
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17,  beta2 = 0.25, beta3 = 0.02),
```

```
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .01)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1, ylim = c(0, 1))
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
for(i in 1:1000){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 =  0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm  ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 =  -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) #0.505


#########
#THETA-3#
##########################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
```

```
}
##########################################


#################
#sample-size 100#
#################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
```

```
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
```

```
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f1)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7,  beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
```

```
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, 1000)
for(i in 1:1000){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = - 6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 =  -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(power = sum(W < 0.05)/N) # 0.684


###########
###########
#  ALPHA  #
###########
###########


N = 1000
mu = 0
sigma = sqrt(.01)

#########
#THETA-1#
#########################################
  f0 = function(x){
pnorm(-(-0.09375 - 0.01875*x))
```

```
}
  f1 = function(x){
  pnorm(-(4.63 + 1.23*x + 0.07*x^2))
}
##########################################


#################
#sample-size 100#
#################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
```

```
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
```

```
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta:4.63 1.23 0.07
#thetahat: -0.09375 -0.01875

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 4.6, beta2  = 1.2, beta3 = 0.06),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 = -0.01),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)
```

```
k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 =  4.3, beta2 = 1.2, beta3 = 0.07),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -0.08, beta2 =  -0.01),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}
(alpha = sum(W < 0.05)/N) #.05


#########
#THETA-2#
############################################
  f0 = function(x){
pnorm(-(-1.484375 - 0.153125*x))
}
  f1 = function(x){
  pnorm(-(0.175 + 0.277*x + 0.024*x^2))
}
############################################


#################
#sample-size 100#
```

```
#################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))


######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
```

```
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
```

```
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: 0.175 0.277 0.024
#thetahat: -1.484375 -0.153125

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = 0.17,  beta2 = 0.25, beta3 = 0.02),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 = -0.15),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
```

```
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#


W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 =  0.17, beta2 = 0.25, beta3 = 0.02),
        algorithm ="default",trace=T)
pop.mod2 = nls(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -1.5, beta2 =  -0.15),
        algorithm ="default", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.046


#########
#THETA-3#
##############################################
  f0 = function(x){
pnorm(-(-5.734375 - 0.5984375*x))
}
  f1 = function(x){
  pnorm(-(-6.69 - 0.60*x + 0.01*x^2))
}
##############################################


##################
#sample-size 100#
##################
n1 = 100

x = seq(-14, -4, by = 1)
w = rep(1/length(x), length(x))
(ns = n1*w)
(ns = round(ns, 0))
(ns = c(10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9))
```

```
######
# f1 #
######

Error1ForPoint1 = matrix(NA, N, ns[1], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[1], mu, 1)
for(j in 1:ns[1]){
Error1ForPoint1[i, j] = E[j]
}
}

Error1ForPoint2 = matrix(NA, N, ns[2], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[2], mu, 1)
for(j in 1:ns[2]){
Error1ForPoint2[i, j] = E[j]
}
}

Error1ForPoint3 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[3], mu, 1)
for(j in 1:ns[3]){
Error1ForPoint3[i, j] = E[j]
}
}

Error1ForPoint4 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[4], mu, 1)
for(j in 1:ns[4]){
Error1ForPoint4[i, j] = E[j]
}
}

Error1ForPoint5 = matrix(NA, N, ns[5], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[5], mu, 1)
for(j in 1:ns[5]){
Error1ForPoint5[i, j] = E[j]
}
}

Error1ForPoint6 = matrix(NA, N, ns[3], byrow = T)
for(i in 1:N){
```

```
E = sigma*rnorm(ns[6], mu, 1)
for(j in 1:ns[6]){
Error1ForPoint6[i, j] = E[j]
}
}

Error1ForPoint7 = matrix(NA, N, ns[7], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[7], mu, 1)
for(j in 1:ns[7]){
Error1ForPoint7[i, j] = E[j]
}
}

Error1ForPoint8 = matrix(NA, N, ns[8], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[8], mu, 1)
for(j in 1:ns[8]){
Error1ForPoint8[i, j] = E[j]
}
}

Error1ForPoint9 = matrix(NA, N, ns[9], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[9], mu, 1)
for(j in 1:ns[9]){
Error1ForPoint9[i, j] = E[j]
}
}

Error1ForPoint10 = matrix(NA, N, ns[10], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[10], mu, 1)
for(j in 1:ns[10]){
Error1ForPoint10[i, j] = E[j]
}
}

Error1ForPoint11 = matrix(NA, N, ns[11], byrow = T)
for(i in 1:N){
E = sigma*rnorm(ns[11], mu, 1)
for(j in 1:ns[11]){
Error1ForPoint11[i, j] = E[j]
}
}

Error1 = cbind(Error1ForPoint1, Error1ForPoint2,
        Error1ForPoint3, Error1ForPoint4, Error1ForPoint5,
```

```
        Error1ForPoint6, Error1ForPoint7, Error1ForPoint8,
        Error1ForPoint9, Error1ForPoint10, Error1ForPoint11)
F1 = Error1

y1 = sapply(x, f0)
Y1 = c(rep(y1[1], ns[1]), rep(y1[2], ns[2]), rep(y1[3], ns[3]),
        rep(y1[4], ns[4]), rep(y1[5], ns[5]), rep(y1[6], ns[6]),
        rep(y1[7], ns[7]), rep(y1[8], ns[8]), rep(y1[9], ns[9]),
        rep(y1[10], ns[10]), rep(y1[11], ns[11]))
X = c(rep(x[1], ns[1]), rep(x[2], ns[2]), rep(x[3], ns[3]),
        rep(x[4], ns[4]),rep(x[5], ns[5]), rep(x[6], ns[6]),
        rep(x[7], ns[7]), rep(x[8], ns[8]), rep(x[9], ns[9]),
        rep(x[10], ns[10]), rep(x[11], ns[11]))

for(i in 1:N){
F1[i,] = Error1[i,] + Y1
}      #F1[1,] is the y-value for 1st simulation

##################################
#Fit a model to simulated data 1#
#original theta: -6.69 -0.60 0.01
#thetahat: -5.734375 -0.5984375

K1 = F1[1,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = -6.7,  beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force", trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 = -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1,pop.mod2)

k1 = summary(pop.mod1)
k2 = summary(pop.mod2)
Xs = seq(-14, -4, .1)
re1 = function(X){
pnorm(-(k1$coefficients[1] + k1$coefficients[2]*X +
        k1$coefficients[3]*X^2))
}
re2 = function(X){
pnorm(-(k2$coefficients[1] + k2$coefficients[2]*X))
}
pre.y1 = re1(Xs)
pre.y2 = re2(Xs)
plot(X, K1)
lines(Xs, pre.y1) #Fitted model1#
lines(Xs, pre.y2) #Fitted model2#
lines(Xs, f1(Xs),col = "red") #Original model#
```

```
W = rep(NA, N)
for(i in 1:N){
K1 = F1[i,]
pop.mod1 = nls2(K1 ~ pnorm(-(beta1 + beta2*X + beta3*X^2)),
        start = list(beta1 = - 6.7, beta2 = -0.58, beta3 = 0.01),
        algorithm ="brute-force",trace=T)
pop.mod2 = nls2(K1 ~ pnorm(-(beta1 + beta2*X)),
        start = list(beta1 = -5.6, beta2 =  -0.6),
        algorithm ="brute-force", trace=T)
ANO = anova(pop.mod1, pop.mod2)
W[i] = ANO$Pr[2]
}

(alpha = sum(W < 0.05)/N) #.066
```