Adaptive Design Optimization for Model Discrimination under Model Misspecification

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Arts in the Graduate School of The Ohio State University

By

Yinghao Sun, B.S.

Graduate Program in Psychology

The Ohio State University

2012

Thesis Committee:

Dr. Jay Myung, Co-Advisor

Dr. Mark Pitt, Co-Advisor

Dr. Michael Edwards

Copyrighted by

Yinghao Sun

2012

Abstract

An experiment in cognitive science is often conducted with the goal of discriminating multiple competing models of a cognitive process. Adaptive design optimization (ADO) is a statistical methodology for selecting the values of the critical design variables (e.g., presentation schedule, stimulus structure) to present on each experimental trial based on responses from the preceding trials such that the chosen values are most informative in differentiating between models under consideration. Prior to applying ADO, a set of candidate models to be discriminated must be specified. Implicit in ADO is the essential assumption that one of the candidate models generates the data, the assumption that may be violated in practice due to model misspecification in which by definition data are generated by none of the candidate models. Even though under model misspecification, identifying the exact form of the data-generating model may not be possible, in present thesis we asked whether ADO model discrimination would nevertheless choose the candidate model that was most similar to the data-generating model in predictions.

Two candidate models of retention memory, power (POW) and exponential (EXP), were compared in two simulations in which the data were generated by a third, different model, that is, when model misspecification happened. In the first, the candidate model predictions were distinct and thus the model discriminability was high. The data were generated from either of two hyperbolic (HYP) models instead of from any of the

candidate models. We found that ADO model discrimination favored the candidate model that was most similar to the data-generating hyperbolic model in predictions. In the second, the candidate models overlapped in predictions and thus the discriminability was low. The data-generating model, as a weighted mixture of power and exponential functions, behaved more “POW-like” when the weight on power function increased and more ”EXP-like” when the weight on power function decreased. We found that ADO model discrimination more strongly favored candidate model POW when the data-generating model behaved more “POW-like”, and vice versa. Finally, the so-called “similar in predictions”, or the model discrepancy was measured by utilizing the Kullback-Leibler divergence, which measures discrepancy between two probability distributions. In summary, in two simulations of model misspecification, ADO model discrimination succeeded in favoring the candidate model that was most similar to the data-generating model in their predictions.

Dedicated to my parents

Acknowledgments

First and foremost, I would like to express the sincerest gratitude to my advisor, Dr. Jay Myung, for his intellectual guidance and constant support, without which I would not have come this far.

I would like to thank Dr. Mark Pitt for his detailed comments and invaluable suggestions on the thesis. My gratitude also goes to Dr. Michael Edwards, for the insights he has shared.

I am also indebted to Dr. Woojae Kim for his vital suggestions on the design of the simulation studies and Dr. Daniel Cavagnaro for his generous help when I encountered issues in implementing the adaptive design optimization algorithm.

I owe my thanks to Yun Tang and Hairong Gu, my fellow labmates, for their unselfish support and encouragement.

Last but not the least, I wish to thank my parents for their unconditional love, ever since my birth.

Vita

2008 …… B.S. Psychology, Sun Yat-Sen University, China

Fields of Study

Major Field: Psychology

Area of Concentration: Quantitative Psychology

Table of Contents

Abstract ii

Acknowledgments v

Vita vi

List of Tables x

List of Figures xi

Chapter 1: Introduction 1

1.1 Cognitive Modeling 1

1.2 Model Discrimination and Design Optimization 2

1.2.1 Model Discrimination 2

1.2.2 Design optimization 4

1.3 Problem of Model Misspecification 6

Chapter 2: Adaptive Design Optimization (ADO) 8

2.1 Bayesian Framework 9

2.2 Computation Algorithm 10

2.2.1 Sequential Updating 11
2.2.2 Utility Function 14
2.2.3 Utility Maximization 17
2.3 ADO in Psychology 21
2.4 Model Misspecification 23
2.4.1 Statistical Robustness 23
2.4.2 Issue of Model Misspecification 25
2.4.3 Purpose of Present Study 26
Chapter 3: Model Discrepancy Measure 28
3.1 Between Two Models 31
3.2 Between a Single Model and a Family of Models 36
Chapter 4: Simulation Experiments 41
4.1 Simulation with High Model Discriminability 42
4.1.1 Simulation Design 43
4.1.2 Simulation Results 45
4.1.3 Conclusion 51
4.2 Simulation with Low Model Discriminability 53
4.2.1 Simulation Design 55
4.2.2 Simulation Results 56

4.2.3 Conclusion 62

Chapter 5: Discussion and Future Directions 63

5.1 Summary 63

5.2 Discussion and Implications 65

5.3 Limitations 66

5.4 Future Directions 67

References 69

Appendix A: Tables and Figures 75

Appendix B: C++ Code 87

List of Tables

Table 1: The parameter estimates (mean ± standard deviation) of model POW and EXP for different data-generating models 85

Table 2: The proportion of consistency between three different K-L divergence based measure and posterior model probability results.....86

List of Figures

Figure 1: Heat map of candidate model POW 75

Figure 2: Histogram of K-L divergence between HYP1 data-generating model and parametric family of POW models 75

Figure 3: Heat map of candidate model EXP 76

Figure 4: Histogram of K-L divergence between HYP1 data-generating model and parametric family of EXP models 76

Figure 5: Heat maps of candidate model POW and EXP in high model discriminability condition 77

Figure 6: Comparison of K-L divergence between estimated POW and EXP with the data-generating model HYP1 77

Figure 7: Histogram of K-L divergence between HYP2 data-generating model and candidate model POW 78

Figure 8: Histogram of K-L divergence between HYP2 data-generating model and candidate model EXP 79

Figure 9: Comparison of K-L divergence between estimated EXP and data-generating model with POW with data-generating model HYP2 79

Figure 10: Heat maps of candidate model POW and EXP in low model discriminability condition 80

Figure 11: Summary of distributions of expected K-L divergence between different mixture data-generating models and parametric family of POW models 81
Figure 12: Summary of distributions of expected K-L divergence between different mixture data-generating models and parametric family of EXP models.....81
Figure 13: Difference in expected K-L divergence between different data-generating mixture models and candidate model POW and between different data-generating models and candidate model EXP.....82
Figure 14: The posterior model probability of POW for different mixture data-generating models.....83
Figure 15: The mixture data-generating model when weight (w) equals 0 or 1.....84

Chapter 1: Introduction

1.1 Cognitive Modeling

Quantitative modeling of cognitive process, as a formal instantiation of theories, plays a crucial role in psychology and cognitive science. By modeling, psychological theories are to be specific and explicit about their assumptions, descriptions, and predictions. The usage of mathematical language helps to achieve clarification which would complement the conventional verbal theories in psychology.

As theories do not always agree with each other, models also differ in their theoretical assumptions and predictions. For example, to capture recall probability decreasing over time in short-term memory experiment, there are several models in the functional form of power, exponential or hyperbolic, competing with each other and all supported by empirical evidence (Rubin & Wenzel, 1996). In addressing behavior in bandit problems, a sequential decision making task, Win-Stay Lose-Shift model holds the assumption that the next decision is only based on the outcome of the most recent decision, while a different model may hold the opposite assumption that decision depends on outcomes of all previous decisions (Zhang & Lee, 2010). Inevitably, for a single phenomenon there often exist multiple models, representing different assumptions and

theoretical implications, “competing” with each other to be a “better” model. This “model competition”, or in other words, model comparison and discrimination, is instrumental in facilitating the evolvement of models and essentially, theories.

1.2 Model Discrimination and Design Optimization

1.2.1 Model Discrimination

Model discrimination is the process in which, based on some formal criteria, a choice is to be made among a set of competing cognitive models. Model discrimination could be a demanding task, with experimentation and data analysis as its two major components. In studies of cognitive process, experimentation is the long established standard way of data collection, of which the design is an art as well as a sophisticated technique. In planning an experiment, numerous factors need to be taken into account, such as sample size, the choice of independent variables, and so on. These considerations are guided by the goal of the experiment, ranging from testing simple hypothesis to discriminating among a set of complex models. Among all these purposes, in particular, model discrimination could be a challenging task, in which mathematical algorithm may become a helpful complement to researchers’ experience and intuitions. For example, in explaining short-term memory, exponential and hyperbolic retention models differ slightly in their curvature predicting the recall probability decreasing over time. Technically, the problem is further complicated by the intrinsic flexibility of these models in the sense that they are both capable of predicting a wide range of data pattern by taking on different parameter values.

Similarly, data analysis methods become more complicated for model discrimination. In model discrimination, criteria include Akaike information criterion (AIC), Bayesian information criterion (BIC), minimum description length, total entropy (Borth, 1975), and mutual information (Lindley, 1956). None of these criteria are perfect in all aspects, e.g., simple goodness-of-fit measure is not a good indicator of generalizability, because it does not properly take into account the intrinsic flexibility of models (Roberts & Pashler, 2000; Pitt & Myung, 2002). Minimum description length, besides its comprehensiveness and better simulation results, is computationally heavy (Grunwald, 2000; Su, Myung, & Pitt, 2005). It is worth noting that, the degree to which data analysis can achieve its pre-specified goal is constrained by the quality of the data, which depends on the experiment design. Better design may lead to "better" data which provide more relevant information and allow for more reliable inference. The present master thesis project is concerned with optimizing the experiment design, and in particular, some theoretical issues in this optimization process.

Design optimization for model discrimination may not be easily achieved, mainly due to the problem of model mimicry, in which models representing different theoretical assumptions mimic each other in their predictions. The importance of model mimicry is long recognized in the works of mathematical modeling in psychology (e.g., Wagenmakers, Ratcliff, Gomez, & Iverson, 2004).

Model mimicry renders it insufficient to solely increase sample size or experimental trials to increase model discriminability. The reason is that some models, with their complicated functional forms and a wide range of possible parameter values,

are able to fit a large scope of data patterns (Navarro, Pitt, & Myung, 2004). For example, a quadratic function can fit a curve as well as a line; therefore it is capable of mimicking a linear function. To discriminate between a quadratic function and a linear function, simply increasing sample size may be inefficient. We may wish to consider at what levels of the independent variable observations should be taken and different choices lead to different efficiency in discriminating the two models. As an extreme case, if we only sample observations from two levels of the independent variable, we could never discriminate between a quadratic function and a linear function even with infinitely large sample size.

The moral is that model discrimination, a potentially difficult task, may demand well-calibrated experiment design since not every design is equally useful in discriminating between models and therefore the optimality of design plays a crucial role in effective (e.g., the conclusion is more reliable) and efficient (e.g., fewer samples are needed) model discrimination.

1.2.2 Design optimization

Design Optimization (DO) is an algorithm to assist researchers in optimizing the experiment design through formalizing the process of finding the best design into a mathematical problem. It is particularly helpful if the goal of an experiment involves complicated statistical inference such as model discrimination.

Design optimality is a topic that has been studied for a long time, in statistics and in other disciplines. Since the pioneering book *The Design of Experiments* (1935) by Ronald

A. Fisher, experiment design has been formally incorporated into the study of statistics. Many statisticians have contributed to this topic, via both Frequentist and Bayesian approaches. Box and Hill (1967) proposed an experiment design optimization method for physics model discrimination by planning the next experiment based on the results of previous experiments. They have also discussed the optimization of the experiment design for effectively testing the fit of a non-linear model (Box, 1969). Optimizing designs in a sequence of experiments aiming to discriminate between economical models has also been studied (El-Gamal & Palfrey, 1996). Experiment design is a frequently studied topic in statistics (e.g., Atkinson & Donev, 1992; Allen, Liyang, & Schmitz, 2003; Bardsley, Wood, & Melikhova, 1996; Schwaab et al., 2006).

Design optimality is also a well-studied topic in machine learning, where the focus is to find the best design to optimize the performance of the program while minimizing its training cost (Settles, 2010). Also called active learning, it carries the essential idea that in a sequence of mini experiments, subsequent experimental design is optimized by taking advantage of previous experiment outcomes; in this way, a better performance of the program may be achieved (Cohen, Atlas, & Ladner, 1994).

In psychology, optimizing experiment design has been developed and applied in psychophysics (e.g., Kujala & Lukka, 2006, Kuhala, 2010; Lesmes, Jeon, Lu & Dosher, 2006; Lesmes, Lu, Baek, & Albright, 2010). Myung and Pitt (2009) implemented the DO algorithm optimizing the design for model discrimination in two content areas: retention and categorization, where models mimic each other and are difficult to distinguish from one another. Later Cavagnaro and colleagues (2010) developed a sequential extension of

DO, adaptive design optimization (ADO) and applied it in experiments with human participants (Cavagnaro, Pitt, & Myung, 2011). The simulation result shows that DO is a promising tool for model discrimination.

1.3 Problem of Model Misspecification

DO and ADO can only discriminate between a set of candidate models that are pre-specified before data collection. One essential assumption of DO and ADO is that data are generated by one of the candidate models, an assumption that is vunerable and likely to be violated in practice. In fact, as George P. E. Box (1979) says “Models, of course, are never true, but fortunately it is only necessary that they be useful”. In empirical experiments, data are generated by the cognitive process that is beyond the comprehensive capture of any existing models. When the data-generating model differs from any candidate models, the problem of model misspecification arises and the essential assumption of ADO is violated. Under model misspecification, DO and ADO could never identify the data-generating model since all of its candidate models are mis-specified compared to the data-generating model. However, if DO and ADO are relatively robust statistical procedures, even under model misspecification, it should be capable of choosing one of the candidate models that is a “best” approximate for the data, the claim which has not been evaluated before.

Adaptive design optimization (ADO) is the sequential extension of DO, transforming the whole experiments into a series of mini experiments, and optimizing the subsequent design based on previously gained information. The purpose of the present

thesis is to explore and examine the performance of adaptive design optimization (ADO) for model discrimination under model misspecification. Specifically, it attempts to answer the question of whether the result of ADO model discrimination under model misspecification is still theoretically valid and useful in the sense that the "best" approximate candidate model would be favored. It aims at evaluating the robustness of ADO and it is of exploratory nature. The remainder of the thesis proceeds as follows. Chapter 2 introduces the mathematical and algorithmic framework of ADO and its application in cognitive science, discusses the issue of model misspecification, and states the purpose of the present thesis. Chapter 3 discusses a measure for model discrepancy in predictions. Chapter 4 includes two simulation studies in which the performance of ADO under model misspecification is evaluated in different conditions. Chapter 5 concludes the thesis with a summary of major results and some discussion about limitations and future directions.

Chapter 2: Adaptive Design Optimization (ADO)

This chapter consists of two parts: the first part (section 2.1, 2.2, and 2.3) provides an overview of the statistical and algorithmic foundation of adaptive design optimization (ADO) as well as its application in cognitive science; the second part (section 2.4) discusses the issue of model misspecification and states the purpose of the present thesis.

ADO adopts a Bayesian framework, with its computation including three major components: a method of sequential updating, a utility function, and a maximization algorithm. In the Bayesian experimental design framework of DO and ADO, the design is optimized through maximizing some utility function with the maximization algorithm proposed by Müller and his colleagues (2004) and Amzal and his colleagues (2006).

DO searches for a set of optimal designs before data collection. As a sequential extension of DO, adaptive design optimization (ADO) transforms the whole experiment into a series of stages (e.g., each trial may be regarded as a stage) connected via a sequential updating method, and between stages, design for the next stage is optimized based on previous experiment outcomes. In this way, each design is optimized by taking advantage of the information available from previous experiments.

2.1 Bayesian Framework

A Bayesian framework is adopted to the implementation of ADO. First of all, the quantitative models are formulated into Bayesian probability models, of which parameters are treated as random variables with a prior distribution. The problem of optimization is approached by the Bayesian decision theoretic approach, in which the goodness of an experiment is quantified into a utility function, the target function for ADO to maximize. In Bayesian framework, randomness in experiment outcomes and uncertainty in model parameters are both taken into account (Chaloner & Verdinelli, 1995).

Formally, the local utility function is denoted as $u(d,\theta,y)$, where d is the design, θ is the parameter value of the model, y is the observation (e.g., participants' response) . Each design can generate several possible outcomes. Local utility evaluates a design based on a single experiment and a single outcome. Global utility $U(d)$ takes into account all possible outcomes under a design, and it is defined as the expected local utility over parameter θ and observation y:

$$U(d)=\int\left[\int u(d,\theta,y)p(y\mid\theta,d)dy\right]p(\theta)d\theta \tag{2.1}$$

In the context of model discrimination, the global utility is slightly modified and defined as

$$U(d) = \sum_{m=1}^{K} p(m) \left\{ \int \left[\int u(d, \theta_m, y_m) p(y_m \mid \theta_m, d) dy_m \right] p(\theta_m) d\theta_m \right\} \tag{2.2}$$

Since there are K (> 1) models instead of one single model, there has to be distinct parameter θ_m for each model m, and the global utility is attained by summing each model's utility weighted by the corresponding model prior probability $p(m)$. Finally, the optimal design d^* satisfies the following equation:

$$d^* = \arg\max_{d} \{U(d)\} \tag{2.3}$$

The value of d^* is the optimal design that achieves the global maxima of the global utility $U(d)$.

2.2 Computation Algorithm

In this section, we will introduce three major components of the computation algorithm in ADO: a method of sequential updating, some candidate utility functions, and an algorithm for utility maximization. ADO updates the parameter distributions and model probabilities after each stage (thus, sequentially), and between stages optimal design is found by maximizing the utility function with an efficient maximization algorithm.

2.2.1 Sequential Updating

After each experiment stage, the model prior probability $p(m)$ and model parameter prior distribution $p(\theta_m)$ are updated based on all preceding observation $y's$, that is, posterior distributions for parameters and models based on previous n-1 stages are taken as the new prior distributions for the nth trial.

Model prior probability is updated via Bayes factor. Bayes factor of Model M_1 over Model M_2, denoted as K, is defined as

$$K = \frac{\mathrm{P}(Y \mid M_1)}{\mathrm{P}(Y \mid M_2)} = \frac{\int \mathrm{P}(\theta_1 \mid M_1)\,\mathrm{P}(Y \mid \theta_1, M_1)\,d\theta_1}{\int \mathrm{P}(\theta_2 \mid M_2)\,\mathrm{P}(Y \mid \theta_2, M_2)\,d\theta_2} \tag{2.4}$$

This is the simplest case where there are only two models to discriminate. Equal prior model probability is assumed,

$$\mathrm{P}(M_1) = \mathrm{P}(M_2) = 0.5$$

By Bayes' theorem,

$$\mathrm{P}(M \mid Y) = \frac{\mathrm{P}(Y \mid M)\,\mathrm{P}(M)}{\mathrm{P}(Y)}$$

Therefore,

$$K = \frac{\mathrm{P}(Y \mid M_1)}{\mathrm{P}(Y \mid M_2)} = \frac{\mathrm{P}(Y \mid M_1)\,\mathrm{P}(M_1)}{\mathrm{P}(Y)} \times \frac{\mathrm{P}(Y)}{\mathrm{P}(Y \mid M_2)\,\mathrm{P}(M_2)} = \frac{\mathrm{P}(M_1 \mid Y)}{\mathrm{P}(M_2 \mid Y)}$$

With the constraint that model probabilities sum to one,

$$\mathrm{P}(M_1 \mid Y) + \mathrm{P}(M_2 \mid Y) = 1$$

The posterior model probability can be attained by

$$P(M_1 \mid Y) = \frac{K}{K+1};$$
$$P(M_2 \mid Y) = \frac{1}{K+1}; \tag{2.5}$$

This posterior model probability based on outcomes of previous stages is treated as the new model prior probability for the next following stage.

Parameter prior distribution is updated through deducing its posterior distribution via Bayes theorem. Since there are more than one competing models, each parameter posterior distribution is deduced by assuming that its corresponding model is true. To sample from these posterior distributions, we can employ Markov Chain Monte Carlo methods, such as Gibbs sampler and Metropolis–Hastings algorithm (Gelman, Carlin, Stern, & Rubin, 2004).

For Gibbs sampler, first Model 1 is assumed correct, and it consists of J parameters, $\theta = (\theta_1, \theta_2, ...\theta_J)$. Then the iterative algorithm to sample from the posterior joint distribution of θ is at each iteration t, sample θ_j from $p(\theta_j \mid \theta_{-j}^{t\text{-}1}, y)$ for all $j = 1, 2, 3..., J$, where $\theta_{-j}^{t-1} = (\theta_1^t, ..., \theta_{j-1}^{t-1}, \theta_{j+1}^{t-1}, ..., \theta_J^{t-1})$. After repeating this for T iterations, we discard a proportion of early samples and retain the rest, which is an approximate sample from the posterior joint distribution of θ for Model 1. Then this procedure is repeated for other models.

On the other hand, Metropolis-Hasting (M-H) algorithm is a general method for drawing samples from posterior distributions, of which Gibbs sampler can be viewed as a special case. It converges to a target distribution with a rejection/acceptance rule and a random walk process. The M-H algorithm can be described as follows:

1. Draw a starting point θ^0, for which $p(\theta^0 \mid y) > 0$

2. For $t = 1, 2, \ldots$:

(a) Sample a proposed θ^* from a proposal distribution, $q(\theta^* \mid \theta^{t-1})$

(b) Calculate the ratio of the densities,

$$r = \frac{p(\theta^* \mid y) / q(\theta^* \mid \theta^{t-1})}{p(\theta^{t-1} \mid y) / q(\theta^{t-1} \mid \theta^*)}$$

(*c*) Set

$\theta^t = \theta^*$ with probability $\min(r, 1)$

$\theta^t = \theta^{t-1}$ otherwise

If the proposal distribution is symmetric, i.e., $q(\theta_a \mid \theta_b) = q(\theta_b \mid \theta_a)$, the density ratio can be simplified to $r = \frac{p(\theta^* \mid y)}{p(\theta^{t-1} \mid y)}$. Besides, in practice a proportion of beginning samples is often discarded because the convergence only happens after a certain number of iterations.

In the current thesis a hybrid of Gibbs sampler and Metropolis-Hasting algorithm is implemented. The framework is the same as Gibbs sampler except that an M-H rejection rule is used in each iteration. The algorithm could be described as follows:

1. Draw a starting point θ^0, for which $p(\theta^0 \mid y) > 0$

2. In interation $T = t$:

For *jth* parameter:

(a) Sample a proposed θ_j^* from a proposal distribution, $q(\theta_j^* \mid \theta_j^{t-1})$

(b) Calculate the ratio of the densities,

$$r = \frac{p(\theta_j^* \mid \theta_{-j}^{t-1}, y) / q(\theta_j^* \mid \theta_j^{t-1})}{p(\theta_j^{t-1} \mid \theta_{-j}^{t-1}, y) / q(\theta_j^{t-1} \mid \theta_j^*)}$$

(*c*) Set

$\theta_j^t = \theta^*$ with probability $\min(r,1)$

$\theta_j^t = \theta_j^{t-1}$ otherwise

(a), (b), (c) are repeated for $j = 1,\ 2,\ 3,\ ...,\ J$ within each iteration t

2.2.2 Utility Function

Utility function measures the degree to which the pre-specified experiment goal is achieved. For example, in an experiment aiming to discriminate models, utility function that is appropriate reflects the discriminability of these models.

The choice of utility function is arbitrary given that it measures the model discriminability. Here two utility functions are introduced, badness-of-fit based on sum-of-square errors, and mutual information.

(1) Badness-of-fit

The essential idea is that model discrimination is possible when some models fit the data badly while others not. If model A is assumed true, or equivalently, the data/experiment outcome $y_A = (y_{A1}, y_{A2}, ..., y_{AN})$ is generated from Model A, the fit should be as poor as possible when Model B is fitted to this data set by minimizing sum-of-square errors; therefore Model B fails to "pretend" to be the true model and the discriminability between two models can be maximized (Myung & Pitt, 2009). That is, when Model A is assumed to be the true model, the utility to maximize is:

$$u(d, \theta_A, y_A) = \sum_{i=1}^{k} (y_i - prd_{Bi}(\theta_B^*, d))^2 \tag{2.6}$$

θ_B^* is the best fitting parameter of Model B to the data set in terms of least sum-of-square errors, and $u(d, \theta_A, y_A)$ measures the sum-of-square errors when the best fitting Model B is fitted to the data. Maximizing $u(d, \theta_A, y_A)$ is equivalent to maximizing the badness-of-fit. Taking parameter and observation uncertainty into account, utility is defined as

$$\iint u(d, \theta_A, y_A) p(y_A \mid \theta_A, d) p(\theta_A \mid d) dy_A d\theta_A$$

Notice again that this is based on the assumption of Model A being a true model. If Model B is assumed to be true, the utility could be defined in a similar way. After combining these two utilities, the global utility function is defined as:

$$\begin{aligned} U(d) = {} & p(A) \iint u(d, \theta_A, y_A) p(y_A \mid \theta_A, d) p(\theta_A \mid d) dy_A d\theta_A \\ & + p(B) \iint u(d, \theta_B, y_B) p(y_B \mid \theta_B, d) p(\theta_B \mid d) dy_B d\theta_B \end{aligned} \tag{2.7}$$

Where $p(A)$ and $p(B)$ are the model prior probabilities subject to the constraint of summing to one, since there are only two models to distinguish.

(2) Mutual Information

Adopted from information theory, mutual information has been applied in model discrimination (Cover & Thomas, 1991; Cavagnaro et al., 2010). It measures the amount of information a set of data can provide and the global utility based on mutual information is defined as:

$$I(M;Y \mid D) = p(A)\iint p(y \mid \theta_A, d)\, p(\theta_A) \log \frac{p(A \mid y, d)}{p(A)}\, dy d\theta_A + p(B)\iint p(y \mid \theta_B, d)\, p(\theta_B) \log \frac{p(B \mid y, d)}{p(B)}\, dy d\theta_B \tag{2.8}$$

After taking into account outcome randomness and parameter uncertainty, equation (2.8) indicates, on average, the amount of reduction in model uncertainty achieved by a design. An optimal design D^* should target the place where least is known and thereby provides the greatest amount of information and leads to the greatest reduction in model uncertainty.

Re-expressing the utility in Bayes factors, the local utility for model A is

$$u(d, \theta_A, y_A) = \log \frac{p(A \mid y_A, d)}{p(A)} = -\log\left[\, p(A) \times BF_{(A,A)}(y_A) + p(B) \times BF_{(B,A)}(y_A) \right] \tag{2.9}$$

for model B it is,

$$u(d, \theta_B, y_B) = \log \frac{p(B \mid y_B, d)}{p(B)} = -\log\left[\, p(A) \times BF_{(A,B)}(y_B) + p(B) \times BF_{(B,B)}(y_B) \right] \tag{2.10}$$

Where, for example, $BF_{(A,B)}(y_B)$ is the Bayes Factor of model A over model B based on observation y_B.

2.2.3 Utility Maximization

One crucial step of ADO is finding the optimal design d^* that maximizes $U(d)$. Since the target function $U(d)$ is defined in terms of multiple integrals, its maximization is non-trivial and could be computationally demanding. In particular, for DO's sequential extension ADO, a shorter computation time is preferred. Conventional optimization method such as the Newton-Raphson method and the Levenverg-Marquardt method may be inefficient in terms of computation time. Fortunately, recent advances in statistical computing (Müller, Sansó, & De Iorio, 2004; Muller, Berry, Grieve, Smith, & Krams, 2007; Amzal, Bois, Parent, & Robert, 2006) enable the application of ADO.

The maximization algorithm consists of three fundamental components: recasting the utility function into an artificial probability distribution of which the mode corresponds to the optimal design; simulating annealing to facilitate the convergence to the mode; incorporating interacting particle systems for more efficient computation and better convergence.

(1) Utility function as a probability distribution

As before,

$$d^* = \arg\max_{d \in D} \{U(d)\}$$

$$U(d) = p(A)\iint u(d, \theta_A, y_A)\, p(y_A \mid \theta_A, d)\, p(\theta_A \mid d) dy_A d\theta_A$$

$$+ p(B)\iint u(d, \theta_B, y_B)\, p(y_B \mid \theta_B, d)\, p(\theta_B \mid d) dy_B d\theta_B$$

According to Müller and his colleagues (2004)'s insight, if $u(d, \theta, y) \geq 0$, we can define an auxiliary distribution $h(\bullet)$, treating d as a random variable and searching the maxima by locating the mode of marginal density $U(d)$. Multiplying a normalizing constant will transform the original utility into a nonnegative function.

$$h(d, y_A, \theta_A, y_B, \theta_B) = \alpha \left[p(A)u(d, \theta_A, y_A) + p(B)u(d, \theta_B, y_B) \right] \times p(\theta_A, y_A, \theta_B, y_B \mid d) \quad (2.11)$$

where $\alpha > 0$ is a normalizing constant and

$$p(\theta_A, y_A, \theta_B, y_B \mid d) = p(y_A \mid \theta_A, d)\, p(y_B \mid \theta_B, d)\, p(\theta_A)\, p(\theta_B)$$

Marginalizing over $(\theta_A, y_A, \theta_B, y_B)$, it is shown that

$$h(d) = \alpha \iint \iint \left[p(A)u(d, \theta_A, y_A) + p(B)u(d, \theta_B, y_B) \right] \times p(\theta_A, y_A, \theta_B, y_B \mid d) d\theta_A dy_A d\theta_B dy_B$$

$$= U(d)$$

Therefore it is verified that the mode of marginal distribution $h(d)$ corresponds exactly to the maxima of global utility $U(d)$. Hitherto, the question of optimization of $U(d)$ becomes how to efficiently find the mode of the marginal distribution $h(d)$.

(2) Simulated annealing

The marginal distribution $h(d)$ may be flat, rendering it difficult to find the mode and there may be many locally optimal designs. To overcome this problem, Müller and his colleagues (2004; 2007) and Amzal and his colleagues (2006) applied simulated annealing, an algorithm which gradually lowers the temperature and sharpens the marginal distribution. As the marginal distribution becomes more peaked, samples drawn from the posterior distribution will be more concentrated around the global mode; via these samples, we could approximately locate the mode. In simulated annealing, the $h(\bullet)$ is modified as

$$h_J(d, y_{A1}, \theta_{A1}, ..., y_{AJ}, \theta_{AJ}, y_{B1}, \theta_{B1}, ..., y_{BJ}, \theta_{BJ}) = \alpha_J \prod_{j=1}^{J} \left[p(A)u(d, \theta_{Aj}, y_{Aj}) + p(B)u(d, \theta_{Bj}, y_{Bj}) \right] \times p(\theta_{Aj}, y_{Aj}, \theta_{Bj}, y_{Bj} \mid d) \quad (2.12)$$

Where J is the reciprocal of temperature, that is, $1/T$, which gradually increases and sharpens the distribution.

(3) Interacting particle filters

Local maxima, instead of global maxima, may be found if only one Markov Chain is employed to draw samples from the marginal distribution $h(d)$, since the sampling procedure may be trapped somewhere close to local maxima. To tackle this problem, Amzal and his colleagues (2006) propose an algorithm in which multiple Markov Chains are ran simultaneously, namely, the sequential Monte Carlo method. These different

Markov chains are combined into a parallel-running and interacting particle system with each chain as a particle, eliminated or propagated according to an evolutionary rule.

To sum up, the algorithm employed by ADO is called Resampling-Markov algorithm, whose detail proceeds as follows (Amzal et al., 2006):

1. Initialization. Start at $t=0$, with a sample $(d_i^{(0)})_{i=1,2,...,N}$; then draw $(\theta_{i1}^{(0)}, y_{i1}^{(0)})_{i=1,2,...,N}$ from $p(\theta, y \mid d_i^{(0)})$. Then we have a sample of $(d_i^{(0)}, \theta_{i1}^{(0)}, y_{i1}^{(0)})_{i=1,2,...,N}$. Set $J = J(0) = 1$. N is the number of particles.

2. Reweighting. For each $i = 1, 2, ..., N$,draw additional independent experiments $(\theta_{ij}^{(0)}, y_{ij}^{(0)})_{j=J(0)+1,...,J(1)}$, and calculate the normalized weights for each particle

$$w_i^{(1)} \propto \prod_{J(0)+1}^{J(1)} u(d_i^{(0)}, \theta_{ij}^{(0)}, y_{ij}^{(0)})$$

3. Resampling. Resample $(\hat{d}_1^{(1)}, ..., \hat{d}_N^{(1)})$ from $(d_1^{(0)}, ..., d_N^{(0)})$ according to a multinomial distribution with weights $w_i^{(1)}$.

4. Metropolis-Hastings step. For each $i = 1, 2, ..., N$, draw $\tilde{d}_i^{(1)}$ from proposal distribution $q_{MH}(d \mid \hat{d}_i^{(1)})$ and J independent experiments $(\tilde{\theta}_{ij}^{(1)}, \tilde{y}_{ij}^{(1)})_{j=1,2,...,J}$ from $p(\theta, y \mid \tilde{d}_i^{(1)})$.

Compute the acceptance rates

$$\alpha_i = \min(1, (\tilde{u}_i^{(1)} q_{MH}(\hat{d}_i^{(1)} \mid \tilde{d}_i^{(1)})) / (\hat{u}_i^{(1)} q_{MH}(\tilde{d}_i^{(1)} \mid \hat{d}_i^{(1)})))$$

where $\tilde{u}_i^{(1)} = \prod_{j=1}^{J} u(\tilde{d}_i^{(1)}, \tilde{\theta}_{ij}^{(1)}, \tilde{y}_{ij}^{(1)})$

Set $d_i^{(1)} = \tilde{d}_i^{(1)}$ with probability α_i and $d_i^{(1)} = \hat{d}_i^{(1)}$ with probability $1 - \alpha_i$.

2.3 ADO in Psychology

Design Optimization and ADO are applied in cognitive science recently, especially in psychology. After explorations and efforts on models evaluation and comparison (e.g., Su et al., 2005; Myung, Navarro, & Pitt, 2006; Pitt & Myung, 2002; Pitt, Myung, & Zhang, 2002), Myung and Pitt and their colleagues came to the insight that instead of solely focusing on data analysis after data collection, it is also useful to consider increasing the quality of the data with improved experiment design. The inquiry begins by finding the right approach (here the Bayesian decision theoretic approach is taken) to formulate the design problem into a multi-variable integration maximization problem, and in the end the method of optimization is found in statistics (Müller, Sansó, & De Iorio, 2004; Muller, Berry, Grieve, Smith, & Krams, 2007; Amzal, Bois, Parent, & Robert, 2006), modified, and applied in psychology (Myung & Pitt, 2009).

So far, DO/ADO has been applied in various studies. For example, Yun and colleagues (2009) applied DO to number line presentation problem, trying to decide which model, linear or logarithmic, could be a better account of a child's inner psychological representation of numbers. Cavagnaro and colleagues (2010) developed a sequential version of DO, that is, ADO, by adopting mutual information as the utility function and optimizing the next design based on all previous experiment outcomes. Zhang and Lee (2010) successfully applied DO to discriminating several sequential decision-making models, that is, models for bandit problems. Besides simulation-based studies, ADO is applied to real participants, for example, Cavagnaro, Pitt, and Myung

(2011) collected data from participants with the experiment design optimized by ADO and based on this data several retention models were compared.

In the field of psychology, there are also many other independent efforts in optimizing experiment designs. Kujala and Lukka (2006) have developed an efficient method for estimating the psychophysics threshold in multi-dimensional psychometric functions by optimizing multi-dimensional stimuli for maximum expected information gain in each trial. The framework has later been extended to take into consideration the experimental cost, e.g., observation time, money etc. (Kujala, 2010; Kujala, Richardson, & Lyytinen, 2010). Similar idea of Bayesian adaptive sequential estimation and optimizing design by maximizing expected information gain has also been developed and applied in estimating the visual contrast threshold in psychophysics vision studies (Lesmes, Jeon, Lu, & Dosher, 2006; Lesmes, Lu, Baek, & Albright, 2010).

So far we have introduced the theoretical and algorithmic foundations of ADO, as well as its applications, particularly in model discrimination. However, one of the essential assumptions held by ADO is that data are generated by one of the candidate models, an assumption that is likely to be violated in empirical experiments. When the data-generating process is different from any candidate models, that is, by definition when model misspecification happens, it remains a question whether ADO for model discrimination could choose the candidate model that is a “best” approximate for the data.

2.4 Model Misspecification

This section provides a brief introduction to the concept of robustness in statistics, discusses the issue of model misspecification in ADO, and states the purpose of the present thesis.

2.4.1 Statistical Robustness

Statistical inference procedures, in particular parametric approaches, have to be based on certain distributional and model assumptions, which however, represent only convenient mathematical approximations to the true underlying situation. Statisticians are concerned with the consequences of assumption violations on statistical inference procedures, of which the optimality may depend on the truthfulness of the original assumption. One of the early contributors to the idea of "robustness" is John W. Tukey, who has pioneering work in four aspects: conceptual, tools, techniques, and procedures (Huber, 2002). The issue of robustness prevails every aspect of statistical inference, such as parameter estimation, hypothesis testing, and so on. Casella and Berger (2001) provide a simple and elegant example, which we discuss below.

Suppose that the random variables $X_1, X_2, ..., X_n$ are independent and follow the normal distribution with mean μ and variance σ^2. One unbiased estimator is the sample mean $\bar{X}$, with $Var(\bar{X}) = \sigma^2 / n$, attaining the Cramer-Rao Lower Bound. There may be small deviations from the distributional assumption, for example, when data are actually generated from a *δ-contamination* model defined as below:

$X_i \sim normal\left(\mu, \sigma^2\right)$ with probability $1-\delta$ and $X_i \sim f\left(x\right)$ with probability δ,

where $f\left(x\right)$ is some other distribution and δ $(0 < \delta < 1)$ is some small value.

This situation poses challenges for the robustness of sample mean as an estimator. Again, as shown by Casella & Berger (2001), under most circumstances the sample mean is still near optimal; however, when $f\left(x\right)$ happens to be the Cauchy distribution, it turns out that $Var(\overline{X}) = \infty$, forbidding us from performing hypothesis testing or confidence interval estimation.

Huber (1981, Page 5) summarized the 'robustness' of a statistical procedure as consisting of the following three desirable features:

"(1) It should have a reasonably good (optimal or nearly optimal) efficiency at the assumed model.

(2) It should be robust in the sense that small deviations from the model assumptions should impair the performance only slightly……

(3) Somewhat larger deviations from the model should not cause a catastrophe."

Huber (1981), in his book, *Robust Statistics*, also discusses in detail the issue of robustness in point estimation, regression, hypothesis testing, etc.

In applied areas, attention has been increasingly drawn to robustness. For example, Farcomeni and Ventura (2012) reviewed several robust inferential procedures that have the potential to become routine practice in biomedical research. In psychology, especially in psychometrics, for example, Wainer (1976) discussed the vulnerability to outliers of mean, standard deviation, produce moment correlation, t-test, and analysis of variance.

Recently, robustness is also an active research topic (e.g., Fan & Hancock, 2011; Magis & De Boeck, 2011).

2.4.2 Issue of Model Misspecification

Prior to applying ADO, a set of candidate models to be discriminated must be specified. Implicit in ADO is the essential assumption that one of the candidate models generates the data. This assumption, however, is likely to be violated in practice. To be a reliable method for model discrimination, the robustness of ADO needs to be evaluated under model misspecification, that is, by definition when the essential assumption of ADO has been violated in the sense that none of the candidate models generate the data. This is, in particular, relevant to the second feature of a robust statistical procedure summarized by Huber (1981) and defined earlier, paraphrased and modified in the current context as: the performance of ADO should not be gravely impaired when its assumption has been moderately violated.

The ADO-chosen model could be interpreted in two ways: either the model that generates the data; or the model that represents the best approximate to the regularity underlying the data in some defined sense. The first one may be doubtful and unrealistic because it is built on the strong assumption that the data-generating model must be one of the candidate models, which, however, is unlikely to be the case. In fact, as George P. E. Box (1979) has said “Models, of course, are never true, but fortunately it is only necessary that they be useful”, it is reasonable to expect that when data are collected from human participants, no existing models can fully capture the underlying cognitive

mechanism. Since it is unlikely that the data generating process is exactly equivalent to that specified by any candidate models, ADO, in particular when applied to human experiments, would constantly face the problem of model misspecification. On the other hand, the second interpretation of the "best" approximate model could still survive under model misspecification, which however, has not been looked into before in the literature. Although the "optimal" design provided by ADO applies when there is no model misspecification, it remains a question of theoretical interest whether ADO is still effective under model misspecification. In sum, despite the fact that application of ADO has been well demonstrated in the past, questions remain for its robustness against the model misspecification problem.

In summary, ADO can only choose among a pre-specified set of candidate models, while on the other hand, the data-generating model may differ from any of the candidate models. With this scenario frequently present in practice, it is necessary to examine the behavior of ADO under model misspecification. This is one major focus of the present thesis.

2.4.3 Purpose of Present Study

Suppose that the candidate models are well supported in the literature and unlikely to be grossly wrong for the true underlying regularity, while the data-generating model still differs from any pre-specified, candidate models in ADO. Under this minor to moderate degree of model misspecification, the key theoretical questions we are interested in answering in this thesis are: (1) is ADO still capable of finding the "best"

approximate model in some defined sense? (2) how reasonable is the result of ADO model discrimination would be in quantitatively responding to the change in data-generating model? Exploratory study of these questions would help us gain insights of the robustness and reliability of ADO under model misspecification.

This issue of model misspecification in ADO has not been investigated before, at least, not in psychology. As long as there is formal quantitative modeling in psychology, we cannot avoid the dilemma that all our models could be wrong, yet being aware of this, modeling could still serve as facilitation to scientific progress in psychology and cognitive science. Since model misspecification is essentially unavoidable, evaluating the robustness of ADO for model discrimination is necessary and would provide a needed assurance to support the usefulness of ADO in human experiments; otherwise the interpretability of ADO would be limited to idealistic but unrealistic settings.

Chapter 3: Model Discrepancy Measure

This chapter discusses how to utilize K-L divergence to measure model discrepancy, and in particular, how this measure could be useful in studying the behavior of ADO under model misspecification.

To monitor and manipulate the degree of model misspecification, it is necessary to first be able to measure the discrepancy between two models in their predictions. Only then may it be possible to know and further manipulate the "difference" between the true model and the misspecified model. Models could be compared in many aspects, such as their functional forms, number of parameters, and so on. However in the present thesis we are only concerned with measuring the "distance" or discrepancy in model predictions.

In the current investigation, simulations will be focused on retention models, which attempt to capture the participants' response in short-term retention memory experimental paradigm. In this paradigm, participants are first asked to memorize n items. After a short time interval, denoted by t, their memory is tested and the number of successfully recalled or recognized items is denoted as y. In this context, time interval (t) is the design variable and number of items recalled (y) is the observation.

The number of recalled items is assumed to follow a binomial distribution. Memorization of each item is independent of one another. Formally, $y \sim binomial(n, p)$, where n is the number of items studied, and p denotes the probability of successfully recalling an item. A memory retention model specifies how the recall probability (p) changes over time. There are several competing retention models, and in the current thesis we will concern ourselves with three of them as follows:

$$POW: p = a(t+1)^{-b}$$
$$EXP: p = ae^{-bt}$$
$$HYP: p = \frac{1}{a+bt}$$

In the above equations, p $(0 \le p \le 1)$ is the recall probability, t is the retention interval between study and test phases and it is the design variable to be optimized in ADO model discrimination algorithm; and finally, a and b are model parameters (For POW and EXP: $0 < a < 1,\ 0 < b < 1,$ For HYP: $1 < a,\ 0 < b$), which, in a Bayesian view, could be viewed as random variables following some prior distribution $\pi(a,b)$.

Given an observation y, the likelihoods of the three retention models take the following forms:

$$For\ POW: L(a,b \mid y) = \binom{n}{y}\left(a(t+1)^{-b}\right)^{y}\left(1-a(t+1)^{-b}\right)^{n-y}$$
$$For\ EXP: L(a,b \mid y) = \binom{n}{y}\left(ae^{-bt}\right)^{y}\left(1-ae^{-bt}\right)^{n-y}$$
$$For\ HYP: L(a,b \mid y) = \binom{n}{y}\left(\frac{1}{a+bt}\right)^{y}\left(1-\frac{1}{a+bt}\right)^{n-y}$$

where $y = 0,1,...,n$.

Each of these models predicts a recall probability (p) at some time interval (t), while each recall probability (p) along with the binomial sample size (n) defines a binomial distribution. Hence, two different models predict two different recall probabilities at a time interval, resulting in two different binomial distributions. Discrepancy between these two binomial distributions (or probablity distributions in general) could be measured based on affine geometry, as suggested by study in the relationship between differential geometry and statistics (Amari & Nagaoka, 2000). One such choice could be Kullback-Leibler divergence (Kullback & Leibler, 1951), which measures the difference between two probability distributions P and Q, defined as follows:

$$D_{KL}(P \| Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \tag{3.1}$$

where P is considered as the true distribution while Q is viewed as a theory, a description or an approximation of P. In the context of present study, Q is the candidate model in ADO, pre-specified and to be discriminated, while P is the data-generating model. However, since at each specific time interval each model defines a binomial distribution, for a set of possible time intervals, each model in turn defines a series of binomial distributions. In the next section we will discuss how to utilize K-L divergence to measure discrepancy in predictions between two retention models with fixed parameter values.

3.1 Between Two Models

In this section, we will discuss how to measure the discrepancy in predictions between two retention memory models with fixed parameter values. Suppose the data-generating model P is $HYP: p = \frac{1}{a_0 + b_0 t}$, where a_0 and b_0 are some fixed values.

However, $POW: p = a_1 (t+1)^{-b_1}$, is an approximate model Q. Between these two models the discrepancy in predictions needs to be quantified.

Note that the model specifies the relationship between recall probability (p) and time interval (t), which together with the binomial distributional assumption, specifies a probability density function. To take into account multiple observations obtained at multiple time intervals, the probability density/ mass function is said to be augmented when it includes more than one observation. Through this augmented probability density/mass function, the model, here HYP or POW, accounts for the data. When the K-L divergence between these two models is based on the augmented probability distributions, it measures the model discrepancy between HYP and POW at multiple time intervals, or a vector of time intervals.

Suppose in memory tests, k observations are taken at k time interval $t's$, where $t = (t_1, t_2, ..., t_k)$, then there are two augmented probability mass functions:

One is based on HYP, with the augmented probability mass function as

$$augmented\ HYP: \quad p(\mathbf{y} \mid a, b) = \prod_{i=1}^{k} \binom{n}{y_i} \left(\frac{1}{a_0 + b_0 t_i} \right)^{y_i} \left(1 - \frac{1}{a_0 + b_0 t_i} \right)^{n - y_i}$$

The other is based on POW, with the augmented probability mass function as

$$augmented\ POW:\ \ q(\mathbf{y}\mid a,b)=\prod_{i=1}^{k}\binom{n}{y_i}\left(a_1\times\left(t_i+1\right)^{-b_1}\right)^{y_i}\left(1-a_1\times\left(t_i+1\right)^{-b_1}\right)^{n-y_i}$$

Distribution P is the true distribution, and distribution Q is the approximation or description for P. As discussed at the beginning of section 3.3, Kullback–Leibler divergence could be used to describe the discrepancy between two probability distributions. It is asymmetric, because it assumes P as the true distribution while Q as an approximation or description of P. This property fits the present scenario of model misspecification. Formally, K-L divergence between augmented HYP and augmented POW is defined as:

$$\begin{aligned}
&D(p(y_1,...,y_k\mid t_1,...,t_k)\,\|\,q(y_1,...,y_k\mid t_1,...,t_k))\\
&=\sum_{(y_1,...,y_k)\in A}p(y_1,...,y_k\mid t_1,...,t_{15})\log\frac{p(y_1,...,y_k\mid t_1,...,t_k)}{q(y_1,...,y_k\mid t_1,...,t_k)}\\
&=E_{p|t_1,...,t_k}\log\frac{p(y_1,...,y_k\mid t_1,...,t_k)}{q(y_1,...,y_k\mid t_1,...,t_k)}
\end{aligned}$$

$$\text{where } A=\{(y_1,...,y_k)\mid y_1=0,1,...,n;\ ...\ y_k=0,1,...,n\}$$

It measures the discrepancy between retention model POW and data-generating model HYP in their predictions at a vector of time intervals. It takes expectation over observation vector $(y_1,...,y_k)$, but it still depends on the value of vector $(t_1,...,t_k)$. To have an indicator of general discrepancy across all possible time interval $t's$, a natural choice is to take the expectation over $t's$. Formally,

$$E_t\left[D(p(y_1,...,y_k \mid t_1,...,t_k) \,\|\, q(y_1,...,y_k \mid t_1,...,t_k))\right]$$
$$= \int_{t\in T}\left[\sum_{(y_1,...,y_k)\in A} p(y_1,...,y_k \mid t_1,...,t_k)\log\frac{p(y_1,...,y_k \mid t_1,...,t_k)}{q(y_1,...,y_k \mid t_1,...,t_k)}\right]\pi(t_1,...,t_k)dt_1...dt_k$$

where $\mathrm{T} = \{(t_1,...,t_k) \mid 0 \le t_1 \le t_{\max};\ ...\ 0 \le t_k \le t_{\max}\}$
and $A = \{(y_1,...,y_k) \mid y_1 = 0,1,...,n;\ ...\ y_k = 0,1,...,n\}$

This is referred to as the expected K-L divergence over a vector of time intervals, since it simultaneously takes into account predictions at a vector of time intervals. This measure could be interpreted as the expected model discrepancy in terms of K-L divergence at a vector of time intervals.

If all observations are mutually independent, expected K-L divergence could be simplified to be based on a single time interval instead of a vector of time intervals. The expression is simplified as:

$$E_t\left[D(p(y_1,...,y_k \mid t_1,...,t_k) \,\|\, q(y_1,...,y_k \mid t_1,...,t_k))\right] = k \times E_t\left[D(p(y \mid t) \,\|\, q(y \mid t))\right]$$

The proof goes as follows: given independence,

$$p(y_1,...,y_k \mid t_1,...,t_k) = p(y_1 \mid t_1)p(y_2 \mid t_2)...p(y_k \mid t_k)$$
$$q(y_1,...,y_k \mid t_1,...,t_k) = q(y_1 \mid t_1)q(y_2 \mid t_2)...q(y_k \mid t_k)$$
$$\pi(t_1,...,t_k) = \pi(t_1)...\pi(t_k)$$

Then the expected K-L divergence over a vector of time intervals between the two augmented distributions becomes:

$$
\begin{aligned}
&E_t\left[D(p(y_1,...,y_k \mid t_1,...,t_k) \,\|\, q(y_1,...,y_k \mid t_1,...,t_k))\right] \\
&= \int_{t\in T}\left[\sum_{(y_1,...,y_k)\in A} p(y_1,...,y_k \mid t_1,...,t_k)\log\frac{p(y_1,...,y_k \mid t_1,...,t_k)}{q(y_1,...,y_k \mid t_1,...,t_k)}\right]\pi(t_1,...,t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[\sum_{(y_1,...,y_k)\in A} p(y_1,...,y_k \mid t_1,...,t_k)\log\frac{p(y_1 \mid t_1)p(y_2 \mid t_2)...p(y_k \mid t_k)}{q(y_1 \mid t_1)q(y_2 \mid t_2)...q(y_k \mid t_k)}\right]\pi(t_1)...\pi(t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[\sum_{(y_1,...,y_k)\in A} p(y_1,...,y_k \mid t_1,...,t_k)\log\frac{p(y_1 \mid t_1)}{q(y_1 \mid t_1)} + ... + \sum_{(y_1,...,y_k)\in A} p(y_1,...,y_k \mid t_1,...,t_k)\log\frac{p(y_k \mid t_k)}{q(y_k \mid t_k)}\right] \\
&\times\pi(t_1)...\pi(t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[\sum_{y_1=0}^{n} p(y_1 \mid t_1)\log\frac{p(y_1 \mid t_1)}{q(y_1 \mid t_1)} + ... + \sum_{y_k=0}^{n} p(y_k \mid t_k)\log\frac{p(y_k \mid t_k)}{q(y_k \mid t_k)}\right]\pi(t_1)...\pi(t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[E_{p|t_1}\log\frac{p(y_1 \mid t_1)}{q(y_1 \mid t_1)} + ... + E_{p|t_k}\log\frac{p(y_k \mid t_k)}{q(y_k \mid t_k)}\right]\pi(t_1)...\pi(t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[D((p \mid t_1) \,\|\, (q \mid t_1)) + ... + D((p \mid t_k) \,\|\, (q \mid t_k))\right]\pi(t_1)...\pi(t_k)dt_1...dt_k \\
&= \int_{t\in T}\left[D((p \mid t_1) \,\|\, (q \mid t_1))\right]\pi(t_1)dt_1 + ... + \int_{t\in T}\left[D((p \mid t_k) \,\|\, (q \mid t_k))\right]\pi(t_k)dt_k \\
&= E_t\left[D(p(y_1 \mid t_1) \,\|\, q(y_1 \mid t_1))\right] + ... + E_t\left[D(p(y_k \mid t_k) \,\|\, q(y_k \mid t_k))\right] \\
&= k \times E_t\left[D(p(y \mid t) \,\|\, q(y \mid t))\right]
\end{aligned}
$$

Therefore, the expected K-L divergence over a vector of time intervals is proportional to expected K-L divergence over a single time interval. Put another way, to see how two models may differ in predictions, it suffices to consider the simplest scenario: when only one observation is taken at only one time interval, what would be the expected prediction discrepancy between two models? By averaging all possible observations and all possible time intervals, expected K-L divergence could be obtained and it indicates the model discrepancy in predictions. For simplicity, expected K-L divergence over a single time interval is referred to as “expected K-L divergence”.

Expected K-L divergence is further simplified under binomial distributional assumption as follows:

$$E_t\left[D(p(y\,|\,t)\,\|\,q(y\,|\,t))\right] = \int_{t\in T}\left[\sum_{(y_1,\ldots,y_k)\in A} p(y\,|\,t)\log\frac{p(y\,|\,t)}{q(y\,|\,t)}\right]\pi(t)dt$$

$$= \int_0^{t_{\max}}\left[\sum_{y=0}^{10} p(y\,|\,t)\log\frac{\binom{n}{y}p_t^{\,y}\left(1-p_t\right)^{n-y}}{\binom{n}{y}q_t^{\,y}\left(1-q_t\right)^{n-y}}\right]\pi(t)dt$$

$$= \int_0^{t_{\max}}\left[\sum_{y=0}^{n} yp(y\,|\,t)\log\frac{p_t}{q_t} + \sum_{y=0}^{n}\left(n-y\right)p(y\,|\,t)\log\frac{1-p_t}{1-q_t}\right]\pi(t)dt$$

$$= \int_0^{t_{\max}}\left[np_t\times\log\frac{p_t}{q_t} + \left(n-np_t\right)\times\log\frac{1-p_t}{1-q_t}\right]\pi(t)dt$$

where $p(y\,|\,t)$ denotes the probability mass function of the true model, and $q(y\,|\,t)$ is the probability mass function of the approximate model; and p_t, q_t, are the recall probabilities, respectively, for the true model and the approximate model; its subscript highlights the fact that the recall probability and the corresponding distribution are different for each different time interval t.

If n = 10, the range of t is 0 to 100 seconds, and each time interval is treated equally by assuming a uniform distribution over time interval t, i.e., $\pi(t) = \frac{1}{100}$ *for* $0 < t < 100$, then expected K-L divergence is numerically approximated in the following way:

$$E_t\left[D(p(y\mid t)\,\|\,q(y\mid t))\right]=\int_0^{t_{\max}}\left[np_t\times p(y\mid t)\times\log\frac{p_t}{q_t}+\left(n-np_t\right)\times p(y\mid t)\times\log\frac{1-p_t}{1-q_t}\right]\pi(t)dt$$

$$\approx\frac{1}{1000}\sum_{i=1}^{1000}\left[np_{t_i}\times\log\frac{p_{t_i}}{q_{t_i}}+\left(n-np_{t_i}\right)\times\log\frac{1-p_{t_i}}{1-q_{t_i}}\right]$$

and t_i's are from 0.001 to 99.901 with an increment of 0.1

In practice, models are estimated based on observations that are taken at a limited number of time intervals. On the other hand, expected K-L divergence, although based on a single time interval, takes into consideration all possible values of this single time interval. Nevertheless, the use of expected K-L divergence is justifiable since estimated models may be often (and implicitly) generalized to be true for all possible time intervals instead of only those discrete values of time intervals at which observations are sampled in a specific experiment. Therefore, expected K-L divergence between data-generating model and an estimated model indicates not only the accuracy of estimation but also its generalizability.

3.2 Between a Single Model and a Family of Models

In implementing ADO model discrimination algorithm, a set of candidate models has to been pre-specified. Each of these candidate models is not a retention model with fixed parameter values; instead, its parameters follow a prior distribution. In other words, with some randomness, the candidate model can take on different parameter values and become different models, thus providing different predictions. However, these

predictions are connected in that they all come from a parametric family of models with the same functional form and parameters following some prior distribution.

Specifically, in the specification of candidate models, there are two components: functional form and parameter prior distribution. For example, for candidate model POW, functional form is defined as $POW: p = a(t+1)^{-b}$, and a joint prior distribution $\pi(a,b)$ is assigned for a and b. Functional form and parameter prior distribution together comprise the fundamental components of candidate model POW, which reflects the *a prior* "belief" about the data pattern. This "belief" is the prediction of the candidate model, which consists of predictions from a parametric family of POW models with different parameter values. The frequency of the occurrences of these different parameter values depends on the parameter prior distribution. In other words, the predictions from a parametric family of POW models are considered plausible, within which different predictions have different degree of plausibility. The prediction of candidate model POW in fact contains the predictions from a parametric family of POW models with their parameters following some prior distribution.

For example, if we define $POW: p = a(t+1)^{-b}$, the prior distribution of a and b independent, with $a \sim beta(40,3),\ b \sim beta(2,25)$. As illustrated in Figure 1, each curve is the prediction of a single model with fixed parameter values; with parameter values sampled from the prior distribution, there are predictions of many models and thus many curves, stacking with each other. The darker it is, the more curves are stacked together, and thus the prediction is more likely. This group of "curves" represents the prediction of candidate model POW. Again, here n = 10, the range of t is 0 to 100 seconds

As illustrated in Figure 1, the prediction of candidate model POW corresponds to such a data pattern that bears the psychological interpretation that the immediate recall is high, while the forgetting rate is slow. Generally, the retention memory is predicted to be good.

However, when model misspecification occurs, the data-generating model becomes, for example, $HYP: p = \frac{1}{2.3 + 0.11t}$, what is the discrepancy between this data-generating model and the prediction of candidate model POW? If the parameters of two retention models are fixed, their predictions are fixed and we know for certain the discrepancy in their predictions in terms of expected K-L divergence. On the other hand, if the parameters of one retention model are random, its prediction would be random as well; therefore the discrepancy in model predictions would also become random, where the randomness depends on the prior distribution. Therefore instead of a single value of expected K-L divergence, we could estimate a distribution of expected K-L divergence to describe the probabilistic pattern of prediction discrepancy between a fixed parameter value model and a candidate model with parameters following some prior distribution.

By sampling 10000 curves representing the prediction of candidate model POW and calculating the expected K-L divergence between these predictions and that of the data-generating model, the histogram obtained is shown in Figure 2, which illustrates the estimated distribution of expected K-L divergence.

The specification of candidate model EXP also consists of two components: functional form and parameter prior distribution. For example, when it is defined that

$EXP:\ p = ae^{-bt}$, the prior distribution of a and b is independent, with $a \sim beta(7,11),\ b \sim beta(1,110)$, the prediction of candidate model EXP could be illustrated as in Figure 3. The prediction of candidate model EXP could be interpreted in a way that the data-generating model is assumed to be in the functional form of EXP, and it describes a low immediate recall with medium forgetting rate. Generally, it is predicted that the retention memory is poor. The estimated distribution of expected K-L divergence between data-generating model HYP and the prediction of candidate model EXP is illustrated in Figure 4.

Comparing these two estimated distributions of expected K-L divergence, the second one is more concentrated on smaller values, indicating that the data-generating model is more similar to the candidate model EXP in predictions. This is expected because $HYP: p = \frac{1}{2.3 + 0.11t}$ describes a poor retention memory with low immediate recall and medium forgetting rate, a similar data pattern shared by candidate model EXP. Furthermore, as in Figure 1, the black curve representing the prediction of the data-generating model is distinct from the prediction of candidate model POW; while as in Figure 3, it is similar to the prediction of candidate model EXP.

In the current thesis, the discrepancy in predictions between a candidate model and a data-generating model is measured by an estimated distribution of expected K-L divergence, instead of a single value. Despite this inconvenience, we could nevertheless utilize this distribution of expected K-L divergence to decide which candidate model is more similar to the data-generating model in predictions. It is expected that ADO for

model discrimination would be capable of choosing the candidate model that is most similar to the data-generating model in predictions.

Chapter 4: Simulation Experiments

This chapter presents two simulation studies in which the behavior of ADO was evaluated under the condition of model misspecification. Specifically, two candidate models for retention memory, power (POW) and exponential (EXP), were compared when the simulated observations were generated by a third model, hyperbolic (HYP). ADO cannot output an estimated model in the true functional form (i.e., HYP), but only in the form of one of the candidate models (i.e., POW or EXP). Hence, identifying the data-generating model would not be possible. However, a theoretical question of interest is whether ADO would be able to choose the "best" approximate model in some defined sense. Specifically, the question is whether ADO favors the candidate model whose prediction is most "similar" to that of the data-generating model. In the present thesis, the so-called similarity in model predictions is quantitatively measured by the expected K-L divergence and intuitively illustrated by comparing the psychological interpretations of these model predictions. For example, compared with other candidate models, if a candidate model has a smaller expected K-L divergence from the data-generating model, and the psychological interpretation of the data pattern predicted by this candidate model is more consistent with that of the data-generating model, we consider this evidence

suggesting that this candidate model could be more similar to the data-generating model in predictions.

It is worth noting that, the prediction discrepancy between the candidate model and the data-generating model is not measured by a single value of expected K-L divergence but instead characterized by an estimated distribution of expected K-L divergence. This is because the data-generating model is a model with fixed parameter values; on the other hand, the so-called candidate models involve specification of functional forms as well as parameter prior distributions. Therefore the prediction of a candidate model consists of predictions from a parametric family of models with the same functional form and different parameter values. As a result, we do not know with certainty the prediction discrepancy between the candidate model and the data-generating model. Instead we could only estimate the probability distribution of this prediction discrepancy.

In addition, the more the candidate models overlap in predictions, the more difficult it is to discriminate between them. In the first simulation study, two candidate models hardly overlap in their predictions, while in the second, the overlap increases.

4.1 Simulation with High Model Discriminability

In the present simulation study, the ability of ADO to discriminate mis-specified candidate models is examined under the condition of high model discriminability in which the predictions of candidate models hardly overlap, as shown in Figure 5. To construct the situation of model misspecification, we employ two data-generating models, which are in the functional form of hyperbolic (HYP), other than any of the candidate

models, namely, power (POW) or exponential (EXP). One of these HYP models is specified to be more similar to candidate model POW and the other HYP model to be more similar to candidate model EXP.

Although model discrimination is easily achieved given that the candidate models are distinct in their predictions, this simulation with high model discriminability is necessary as a starting point to evaluate the performance of ADO under model misspecification. In addition, ADO is conducted under the condition of correct model specification in which one of the candidate models actually generates the data, for it is necessary to first confirm that ADO works properly under normal condition (correct model specification).

4.1.1 Simulation Design

There are two candidate models, POW and EXP. The functional form and parameter prior distribution of candidate model POW are defined as: $POW:\ p = a(t+1)^{-b}$, with $a \sim beta(40,3),\ b \sim beta(2,25)$. This model predicts that the retention memory is generally good, with high immediate recall and a slow forgetting rate over time. On the other hand, candidate model EXP is defined as: $EXP:\ p = ae^{-bt}$, with $a \sim beta(7,11),\ b \sim beta(1,110)$. Candidate model EXP predicts relatively poor retention memory, opposite to the prediction of candidate model POW. These two candidate models are the same as in section 3.3.2, and illustrated in Figure 5.

In this simulation, there are two conditions: correct model specification and model misspecification. In correct model specification, data were generated from a model that

had the same form as one of the candidate models. Simulation under this condition is to examine whether ADO algorithm works as intended under normal condition. In model misspecification, data were generated from a model that was different from any of the candidate models. There were two different data-generating models in each of the two conditions.

The correct model specification condition involves the following two data-generating models:

(1) $POW1: p = 0.90(t+1)^{-0.08}$. When data are generated from this model, candidate model POW is correctly specified since it has the same functional form with the data-generating model. This model predicts relatively good immediate recall and a slow forgetting rate, that is, good retention memory in general, which is also similar to the prediction of candidate model POW.

(2) $EXP1: p = 0.34e^{-0.01t}$. When data are generated from this model, candidate model EXP is correctly specified since it has the same functional form. This model predicts a poor retention memory, with low immediate recall and a quick forgetting rate, which is similar to the prediction of candidate model EXP.

The model misspecification condition involves the following two data-generating models:

(3) $HYP1: p = \frac{1}{2.3 + 0.11t}$, the same model as in section 3.3.2, has a functional form that differs from either of the two candidate models (POW and EXP). It predicts a relatively poor immediate recall and a fast forgetting rate, that is, poor retention memory.

(4) $HYP2: p = \frac{1}{1.05 + 0.013t}$, the opposite of HYP1, predicts relatively good immediate recall and a slow forgetting rate, that is, good retention memory.

In the present simulation study, binomial sample size (n) equals 10 and time interval (t) is between 0 and 100 seconds. For each of the four data-generating models, the same simulation experiment was independently repeated 20 times. Each experiment consisted of 10 stages, and in each stage a single observation was taken at a single time interval. Between stages, ADO was employed to find a one-dimensional optimal design for the next stage. The model posterior probability at the end of each experiment was calculated from the Bayes Factor.

4.1.2 Simulation Results

The result section proceeds in the order that for each of the four data-generating models, we first evaluate the prediction discrepancy between the data-generating model and the candidate model. Next, we present the ADO-estimated model posterior probability of each candidate model. Finally, we examine the estimated models.

We begin this section by discussing results from ADO simulations under the condition of correct model specification, in which simulated observations were generated from a model that had the same functional form as one of the two candidate models, power (POW) and exponential (EXP), with their functional forms and parameter prior distributions defined as in section 4.1.1.

We first generated data from a power model defined as $POW1: p = 0.90(t+1)^{-0.08}$. To see which candidate model is more similar to the data-generating model, we look at the estimated distribution of the expected K-L divergence. Since the distribution is likely to be skewed to the right instead of being normal, it is necessary to examine quartiles instead of solely means. The estimated distribution of the expected K-L divergence between the data-generating model (POW1) and candidate model POW was characterized by the following statistics: mean 0.0.5194, median 0.2588, first quartile 0.0634, and third quartile 0.7123. Similarly, an estimated distribution of the expected K-L divergence between POW1 and candidate model EXP had the mean of 4.4014, the median of 3.7221, the first quartile of 2.1821, and the third quartile of 5.8681. The former distribution was more concentrated on smaller values, which was counted as evidence suggesting that data-generating model POW1 be more similar to candidate model POW in predictions than to candidate model EXP. In each of 20 simulation experiments, ADO estimated the posterior model probability of candidate model POW being greater than 0.9929 at the end of each run, with mean 0.9993. The means of the posterior distributions of the scaling (a) and decay (b) parameters were averaged over 20 simulation runs to be 0.9240 and 0.0857, respectively, which were almost identical to the parameter values of POW1. In short, these results, taken together, indicate that ADO performed as intended, that is, correctly identifying the data generating model as well as recovering its parameter values.

We also generated data from an exponential model defined as $EXP1: p = 0.34e^{-0.01t}$. The estimated distribution of the expected K-L divergence between the data-generating

model (EXP1) and candidate model POW was characterized by the following statistics: mean 6.5140, median 6.0541, first quartile 3.9473, and third quartile 8.5530; the estimated distribution of the expected K-L divergence between EXP1 and candidate model EXP had the mean of 0.4822, the median of 0.2567, the first quartile of 0.0892, and the third quartile of 0.6466. The latter distribution was more concentrated on smaller values, which was counted as evidence suggesting that data-generating model EXP1 be more similar to candidate model EXP in predictions than to candidate model POW. In each of 20 simulation experiments, ADO estimated the posterior model probability of candidate model EXP being higher than 0.9999 at the end of each run. The means of the posterior distributions of the scaling (a) and decay (b) parameters were averaged over 20 simulation runs to be 0.3346 and 0.0098, respectively, which were almost identical to the parameter values of EXP1.

Having demonstrated the ability and effectiveness of ADO to identify the data-generating model and recover parameter values under correct model specification, we now turn our attention to ADO under model misspecification—the main purpose of the present study. That is, we conducted a series of ADO simulations under the condition of model misspecification, in which by definition simulated observations were generated from neither of the two candidate models but instead from a third model.

When data were generated from a hyperbolic model defined as $HYP1: p = \frac{1}{2.3 + 0.11t}$, instead of any candidate models POW or EXP, the problem of model misspecification arose. Figure 2 and 4 display the histogram of the expected K-L divergence between the data-generating model (HYP1) and candidate model EXP as well

as between HYP1 and candidate model POW. Apparently, the estimated distribution of the expected K-L divergence between the data-generating model (HYP1) and candidate model EXP was more concentrated on smaller values. Moreover, candidate model EXP predicted a data pattern representing low immediate recall and a fast forgetting rate, a relatively poor retention memory and a similar psychological interpretation that was shared by the prediction of HYP1. Smaller expected K-L divergence and similar psychological interpretations together indicate that candidate model EXP may be more similar to data-generating model HYP1 in predictions than candidate model POW does.

In each of 20 simulation experiments, ADO estimated the posterior model probability of candidate model EXP being higher than 0.9957 at the end of each run, with mean 0.9995, indicating that candidate model EXP was chosen consistently each time.

Now we look at the estimated models. After each simulation experiment, one exponential model and one power model could be estimated. Estimated model EXP is defined as $p = a_b e^{-b_b t}$, where a_b and b_b are Bayes estimators, here chosen to be the mean of posterior distributions. In other words, the values of a_b and b_b, or the estimates of scaling (a) and decay (b) parameters, were the means of posterior distributions for the scaling (a) and decay (b) parameters, respectively. As for estimated model EXP, the parameter estimate for scaling (a) was 0.3254 ± 0.0540 (mean ± standard deviation), and for decay (b) was 0.0163 ± 0.0540. The estimated model EXP mimicked the data-generating model HYP1 in that the immediate recall probability was low while the forgetting rate was high. Similarly, an estimated model POW in the form of $p = a_b (t+1)^{-b_b}$ was obtained. The parameter estimate for scaling (a) was 0.9072 ±

0.0051 (mean ± standard deviation), and for decay (b) it was 0.3918 ± 0.0369. Notably, estimated model POW was constrained in its ability to mimic the data-generating model HYP1 for its scaling parameter (a) was too high. Nevertheless, the estimated model POW still mimicked the data-generating model HYP1 in that its curve quickly decreased to such a low level that was similar to the pattern of HYP1.

In addition, we investigate the relationship between estimated models, expected K-L divergence and ADO estimated model posterior probabilities. If the discrepancy between estimated model EXP and data-generating model HYP1 was smaller as indicated by smaller expected K-L divergence, it was expected that candidate model EXP had a higher posterior model probability, and vice versa. Expected K-L divergences were calculated between estimated model POW and data-generating mode HYP1, as well as between estimated model EXP and data-generating model HYP1. These expected K-L divergences measure the discrepancy in model predictions between data-generating model HYP1 and either of estimated models (POW or EXP).

Since in all simulation experiments candidate model EXP was favored with a higher posterior model probability, it was expected that expected K-L divergence between data-generating model HYP1 and estimated model EXP be smaller. However, as shown in Figure 6, in 3 among 20 simulation experiments, the expected K-L divergence between estimated model EXP and the data-generating model turned out to be larger, contrary to the expectation. In sum, model posterior probabilities calculated from the Bayes factor and the expected K-L divergence between estimated models and the data-

generating model was not entirely consistent --- in a proportion of 85% simulation experiments it was consistent.

Finally, data were generated by a hyperbolic model defined as $HYP2: p = \frac{1}{1.05 + 0.013t}$, which predicts a data pattern representing good retention memory, a psychological interpretation similar to that of candidate model POW. In addition, its expected K-L divergence with candidate model POW was more concentrated on smaller values than that with candidate model EXP, as illustrated in Figure 7 and 8. This evidence suggested that data-generating model HYP2 may be more similar to candidate model POW in predictions.

In each of 20 simulation experiments candidate model POW was favored. ADO estimated the posterior model probability of candidate model POW being greater than 0.6972 at the end of each run, with the mean of 0.9380 and the median of 0.9612.

Again we look at the estimated models. For estimated model POW, the parameter estimate for scaling (a) was 0.9301 ± 0.0055 (mean ± standard deviation), and for decay (b) was 0.1101 ± 0.0201. For estimated model EXP, the parameter estimate for scaling (a) was 0.6431 ± 0.0471 (mean ± standard deviation), and for decay (b) was 0.0028 ± 0.0009. Both estimated model POW and EXP mimicked the pattern of the data-generating model HYP2, that is, high immediate recall probability and slow forgetting rate. However, estimated model EXP was constrained in its ability to mimic HYP2 since its scaling (a) was too low.

Furthermore we explored whether ADO estimated model posterior probabilities and the expected K-L divergence between estimated models and the data-generating

model were consistent. It was expected that when ADO chose candidate model POW, the expected K-L divergence between estimated model POW and the data-generating model was smaller. As shown in Figure 9, in 17 among 20 simulations, that is, 85% proportion of times, it was smaller and the expectation was met. We discuss this minor inconsistency in the following section.

4.1.3 Conclusion

In the first simulation study, we utilized the expected K-L divergence to measure the model discrepancy, investigated the ADO estimated posterior model probabilities, and examined the estimated models.

In studying the problem of model misspecification, model discrepancy measure is necessary. So far we have employed expected K-L divergence, that is, average K-L divergence over possible time intervals, as a useful measurement for the discrepancy in prediction between a data-generating model and another approximate model (either an estimated model or a candidate model). Since the prediction of a candidate model consists of predictions from a parametric family of models with different parameter values, a distribution of expected K-L divergence could be estimated to measure the discrepancy in prediction between the candidate model and the data-generating model.

The ADO algorithm for model discrimination were run under high model discriminability condition in which there were only two candidate models with distinct predictions. Result has shown that even if the data-generating model had a different functional form from any candidate models, that is, model misspecification, ADO was

still capable of choosing the candidate model that yields the most "similar" prediction to that of the data-generating model. Furthermore, the estimated models, both POW and EXP, mimicked the data-pattern of the data-generating models.

When we looked at the relationship between the expected K-L divergence, estimated models, and ADO estimated model posterior probabilities, it was expected that when a candidate model had a higher posterior model probability, the expected K-L divergence between the corresponding estimated model and the data-generating model should be smaller. However, regarding this expectation some minor inconsistency was found. There are three plausible explanations: first, expected K-L divergence is based on averaging K-L divergence over all possible values of time interval, while on the other hand the estimated model is only based on the information obtained from observations at a limited number of values of time interval, therefore the issue of generalizability is involved; second, since the estimated POW and EXP models are based on Bayes estimator, i.e., posterior means, the whole parameter posterior distribution has not been taken into consideration and hence some information would have been lost; third, model posterior probability is calculated via Bayes factor, which may not be mathematically equivalent to expected K-L divergence.

As a conclusion, in this simulation study, expected K-L divergence has been employed as one measurement of model discrepancy. Under high model discriminability condition, ADO is shown to be capable of choosing, among two candidate models, the one that is more similar to the data-generating model in predictions.

4.2 Simulation with Low Model Discriminability

In the second simulation study, the behavior of ADO for model discrimination is examined under model misspecification when two candidate models highly overlap in their predictions, that is, when model discriminability is low. Furthermore, the data-generating model is adjusted along a continuous continuum, of which one end is "EXP-like" and the other end is "POW-like". We evaluate whether ADO estimated posterior model probability is responsive to this adjustment in a reasonable way in that "EXP-like" data-generating model will lead to a higher posterior probability of model EXP, and vice versa.

Before discussing the details of the second simulation study, let us briefly return to the first simulation. That is, it is worth noting that there are a few limitations of the first simulation study. First, the predictions of two candidate models barely overlap so that ADO algorithm may not even be necessary due to the high model discriminability. Second, in the first simulation we have only chosen two different hyperbolic models in model misspecification condition. However, it is of theoretical interest if we could adjust the data-generating model in a more "continuous" manner. Accordingly, in the second simulation study several improvements have been made.

To manipulate model misspecification more continuously, and to place model correct specification and misspecification on a single continuum, we have employed a power – exponential (POW-EXP) mixture model as the data-generating model defined as:

$$p = w \times a_{POW} \left(t+1\right)^{-b_{POW}} + \left(1-w\right) \times a_{EXP} e^{-b_{EXP} t}$$

Where a_{POW}, b_{POW}, a_{EXP}, and b_{EXP} are fixed parameter values, and $0 \le w \le 1$. When $w = 1$, the data-generating model is a POW function; when $w = 0$, the data-generating model is an EXP function. Therefore when $w = 1$ or $w = 0$, either candidate model, EXP or POW, is correctly specified. When $0 < w < 1$, the data generating model is a mixture of POW and EXP functions, different from any candidate models and resulting in model misspecification. The closer w is to 1, the more the data-generating model behaves "POW-like", and the closer w is to 0, the more the data-generating model behaves "EXP-like". On the single, continuous continuum of $0 \le w \le 1$, not only correct specification and misspecification are connected, but a precise control of the extent to which the data-generating model is different from either of the candidate model EXP or POW could also be achieved.

Furthermore, by changing the prior distribution of model parameters, two candidate models could highly overlap in predictions, resulting in low model discriminability and ADO becoming necessary.

Besides these major changes, one minor change is relevant to how to calculate the expected K-L divergence between estimated POW or EXP model and the data-generating model. Previously we only concern ourselves with expected K-L divergence which is averaged over all possible time intervals. In addition to this "global" K-L divergence, we can average the K-L divergence only across those time intervals at which the observations are obtained in a specific experiment, which could be considered as "local" K-L divergence. "Local" K-L divergence involves less issues of extrapolation or

generalizability from which "global" K-L divergence suffers. In the second simulation, both "global" and "local" K-L divergences are included.

4.2.1 Simulation Design

There are two candidate models, POW and EXP. The functional form and parameter prior distribution of candidate model POW is defined as: $POW: p = a(t+1)^{-b}$, with $a \sim beta(2,1),\ b \sim beta(1,4)$. Candidate model EXP is defined as: $EXP: p = ae^{-bt}$, with $a \sim beta(2,1),\ b \sim beta(1,80)$. These two candidate models highly overlap in their predictions, as shown by the heat map in Figure 10.

The data-generating model is defined as $p = w \times 0.833(t+1)^{-0.391} + (1-w) \times 0.833 e^{-0.0223t}$, where $w = 0,\ .1,\ .2,\ \ldots,\ .9,\ 1$; therefore in total there are 11 different data-generating mixture models. Each model is a weighted mixture of a power (POW) function and an exponential (EXP) function.

When $w = 0$, the data-generating model is an EXP function, therefore candidate model EXP is correctly specified. When $w = 1$, the data-generating model is a POW function, therefore in this case candidate model POW is correctly specified. Under these two cases of model correct specification, the performance of ADO is first checked to ensure that the algorithm works as intended.

For each of the five mixture data-generating models with $w = 0,\ 0.1,\ 0.2,\ 0.9,\ 1$, the same simulation experiment was independently repeated 10 times. Each experiment was consisted of 10 stages, in each stage observation taken at a single time interval. For

each stage ADO had to find a one-dimensional optimal time interval. For each of the other six mixture data-generating models with $w = 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7, 0.8$, the same simulation experiment was independently repeated 20 times. There are more repetitions for these mixture models in order to reduce extra variability found in preliminary results. Each experiment was consisted of 10 stages, in each stage observation taken at a single time interval.

4.2.2 Simulation Results

In this section, we first examine the performance of ADO for model discrimination under correct model specification. Next, we check whether the adjustment of the data-generating model (by changing the weight (w)) does succeed in controlling the "distance" between the data-generating model and the two candidate models through examining the change in the expected K-L divergence. As the main purpose of this simulation study, we examine whether the ADO-estimated posterior model probability responds to the adjustment of the data-generating model in a reasonable way. Finally, we look at the estimated models and explore the relationship between posterior model probabilities, expected K-L divergence, and estimated models.

We begin this section by discussing results from ADO simulations under the condition of correct model specification, that is, by definition when data were generated from one of the candidate models. When $w = 0$, the data-generating model, which became an exponential (EXP) function, was defined as $p = 0.833e^{-0.0223t}$. As shown in Figure 11, the first quartile, the median, the mean, and the third quartile of the estimated

distribution of expected K-L divergence between the data-generating model and candidate model POW were the largest, while as shown in Figure 12 these statistics of the estimated distribution of expected K-L divergence between the data-generating model and candidate model EXP were the smallest. These indicate that when $w = 0$ the exponential data-generating model was most similar to candidate model EXP, which shares the same functional form. In 10 simulation runs, ADO estimated the posterior model probability of candidate model EXP being greater than 0.8800, with mean 0.9824 and median 0.9956. As before, posterior means were taken as parameter estimates. In these 10 simulations, the parameter estimate of scaling (a) was 0.7899 ± 0.0862 (mean standard ± deviation), and that of decay (b) was 0.0201 ± 0.0062, which were almost identical to the parameter values of the data-generating model.

When $w = 1$, the data-generating model, which became a power (POW) function, was defined as $p = 0.833(t+1)^{-0.391}$. As shown in Figure 11 and 12, when $w = 1$, the data-generating model was most similar to candidate model POW. In 10 simulation runs, ADO estimated the posterior model probability of candidate model POW being greater than 0.8245, with mean 0.9611 and median 0.9876. As before, posterior means were taken as parameter estimates. In these 10 simulations, the parameter estimate of scaling (a) was 0.8259 ± 0.0940 (mean standard ± deviation), and that of decay (b) was 0.3580 ± 0.0746, which were almost identical to the parameter values of the data-generating model.

In sum, it had been shown that ADO succeeded consistently in identifying the data-generating model with information only from 10 experimental stages, that is, 100

bernoulli trials in total (because each experimental stage was a binomial trial of sample size 10). Hence, the effectiveness of ADO under correct model specification had been demonstrated. Now we turn our attention to the performance of ADO for model discrimination under model misspecification—the main purpose of the present study. That is, we conducted a series of ADO simulations under the condition of model misspecification, which happens when the data-generating model is neither a POW nor an EXP model, with weight (w) between 0.1 and 0.9.

First, we check whether the manipulation of the mixture weight resulted in an expected change in the discrepancy between the data-generating model and the two candidate models as measured by expected K-L divergence. Distributions of expected K-L divergence between each data-generating model and either of the two candidate models were estimated. The exact method of calculation was in the way as described in section 3.3.2, with a Monte Carlo sample size of 20000. Figure 11 and Figure 12 display the change of mean, median, first and third quartile of the estimated distributions of expected K-L divergence. The mean, median, first and third quartile of expected K-L divergence between mixture data-generating model and candidate model EXP became smaller as w decreased, indicating that when the data-generating mixture model behaved more "EXP-like", it became more similar to candidate model EXP. The pattern of change was similar for expected K-L divergence between mixture data-generating model and candidate model POW, in that the median decreased while w increased. There was one exception that the third quartile and mean started to increase approximately after $w = 0.6$. The reason is that when $w = 1$, the power function did not lie exactly at the center of the

predictions of candidate model POW; therefore while w increased, the mixture model could become further away from some predictions "on the boundary". Nevertheless, the result suggests that in general, when the mixture model behaved more "POW-like" with larger w, it became more similar to candidate model POW.

Additionally, as shown in Figure 13, difference in expected K-L divergence from mixture data-generating model to candidate model POW and to candidate model EXP decreased from positive to negative as w increased. This suggests that while w increased, the mixture data-generating model became more similar to candidate model POW and less similar to candidate model EXP, and vice versa. In summary, the manipulation of weighted mixture model succeeded in controlling its "distance" with candidate model POW and EXP.

As the main purpose of this simulation study, we examine whether the ADO-estimated posterior model probability responded to the change of the data-generating model as predicted. Figure 14 summarizes the posterior model probability of candidate model POW when different mixture models generated the data. When w was not equal to either 0 or 1, both candidate models were mis-specified. As the weight (w) increased, the posterior model probability of candidate model POW increased. In other words, when the prediction of the data-generating mixture model became more similar to that of candidate model POW as measured by expected K-L divergence, the posterior model probability of candidate model POW also increased accordingly, and vice versa. Note that the median of posterior model probability of candidate model POW was almost monotonically increasing in weight (w) with the exception that the median was smaller at $w = 0.8$ than

that at $w = 0.7$. However, the posterior model probabilities of candidate model POW at $w = 0.7$ and $w = 0.8$ overlapped to such a large extent that no significant difference could be expected. The correlation between posterior model probability of candidate model POW and w was 0.706, with $t_{168} = 12.9317$, significant at 0.001 level. In addition, as shown in Figure 14, the interquartile range of posterior model probability of POW was larger for $w = 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$, and smaller for $w = 0, 0.1, 0.2, 0.9, 1$. The larger the interquartile range is, the more uncertainty there is. This result agrees with the intuition: when w is close to neither 0 nor 1, the mixture model was similar to neither candidate model POW nor EXP; therefore no obvious choice could be made between these two candidate models and the uncertainty turned out to be larger.

Finally, we look at the estimated models and explore the relationship between posterior model probabilities, expected K-L divergence, and estimated models. After each simulation experiment, there were posterior distributions for each model parameters. Estimated models were obtained by taking the posterior mean as the parameter estimates. Parameter estimates for POW and EXP under each of the mixture data-generating models are presented in Table 1. These estimates were discussed with regards to the corresponding mixture data-generating model which, as the weight (w) increased, continuously moved from the red curve to the black curve as shown in Figure 15.

The parameter estimate of scaling (a) in model POW was relatively unchanged, because the starting point did not change as weight (w) changed, as shown in Figure 15. The parameter estimate of decay (b) in model POW increased as weight (w) increased, because the “curve” decreased more quickly over x-axis as shown in Figure 15. For

model EXP, the parameter estimate of scaling (a) decreased as weight (w) increased. One plausible explanation was that the exponential curve had to decrease its scaling (a) parameter to stay at a lower level, so as to mimic the black "curve". The estimate of decay (b) parameter in model EXP was relatively unchanged. This result shows that power model and exponential model accommodated the data in a different way. However, we have yet to find a reliable, quantitative method to evaluate whether these estimates are good or bad in a more absolute sense.

Last but not the least, we explore the relationship between posterior model probabilities, expected K-L divergence, and estimated models. Between either of the estimated models and the corresponding data-generating model, K-L divergence based measure could be calculated in two ways: first, average K-L divergence over all possible time intervals, namely, "global" K-L divergence; second, average K-L divergence over only those ten time intervals at which observations were taken in the simulation experiment, namely, "local" K-L divergence. It was expected that when the estimated POW model was more similar to the data-generating model in predictions as measured by global or local K-L divergence, the posterior model probability of POW should be higher, and vice versa. As shown in Table 2, the consistency between comparison of local K-L divergence and ADO model posterior probability result was higher than that of global K-L divergence, with both at an acceptable level.

4.2.3 Conclusion

The second simulation experiment has demonstrated that ADO for model discrimination under model misspecification was still "meaningful" in that it responded to the change of the data-generating model in a quantitatively reasonable and predictable way. Specifically, when the data-generating mixture model was more similar to candidate model POW in predictions as measured by expected K-L divergence, the posterior model probability of candidate model POW increased, and vice versa. Therefore even though ADO model discrimination could not exactly identify the data-generating model when candidate models were mis-specified, the posterior model probability was nevertheless an indicator of whether the data-generating model was more similar to one candidate model or another in their model predictions. However, the posterior model probability solely may not serve as an indicator of whether model misspecification actually occurred.

Admittedly, even though model estimates were intuitively reasonable, we have yet to find a reliable method to evaluate the "accuracy" of these estimates. The comparisons of global K-L divergence and local K-L divergence between estimated models and the data-generating model did not always agree with posterior model probability results because these procedures were not mathematically equivalent as discussed in section 4.1.3. On the other hand, local K-L divergence based on only those ten time intervals used in the experiment resulted in higher consistency, for less generalizability was involved. Questions remain how to best utilize K-L divergence measure.

Chapter 5: Discussion and Future Directions

5.1 Summary

In the present thesis, the robustness of ADO has been evaluated under model misspecification in two simulation studies with different levels of model discriminability. The term "robustness" has many interpretations and in the thesis we specifically examine the second feature of a robust statistical procedure as summarized by Huber (1981), that is, whether the performance of the ADO algorithm for model discrimination would be impaired under small deviations from the ADO assumption that one of the candidate models generates the data. In addition, expected K-L divergence has been employed as a useful measure for model discrepancy in predictions.

First, provided with a set of candidate models to discriminate, ADO would search for a set of optimal designs to best discriminate between the candidate models. The chosen candidate model has two possible interpretations: it is the data-generating model; or it is a "best" approximate for the data in the sense that its prediction is most similar to that of the data-generating model. The first interpretation is prone to failure under model misspecification when ADO assumption is violated and none of the candidate models generate the data. However, as shown in the present thesis, the latter interpretation is still

valid. In other words, under model misspecification, ADO favors the candidate model that is most similar to the data-generating model in predictions.

In the first stimulation, the prediction of candidate model POW corresponds to a data pattern representing good retention memory, and the prediction of candidate model EXP corresponds to a data pattern representing poor retention memory. Two candidate models hardly overlap in predictions so that the model discriminability is high. The data-generating models, hyperbolic (HYP) functions, differ from any candidate models. ADO model discrimination favors candidate model POW if the data-generating model has a similar prediction corresponding to a data pattern with high immediate recall and a slow forgetting rate, and favors candidate model EXP if the data-generating model has a prediction corresponding to a low immediate recall and fast forgetting rate data pattern. In summary, in this simulation when none of the candidate models generates the data and ADO assumption has been violated, ADO model discrimination would still favor the candidate model that is most similar to the data-generating model in predictions. In the present thesis, the so-called similarity in prediction is quantitatively measured by the expected K-L divergence and intuitively examined with regards to the psychological interpretations of the data patterns these models predict.

In the second simulation, two candidate models, POW and EXP, highly overlap in predictions so that the model discriminability is low. The data-generating model is a weighted mixture of a power function and an exponential function. As shown by the result, ADO for model discrimination more strongly favors candidate model POW if the mixture data-generating model behaves more “POW-like” with larger weight on the

power function, and vice versa. This suggests that ADO for model discrimination is responsive to the underlying change of the data-generating model in a quantitatively reasonable way.

K-L divergence (Kullback & Leibler, 1951) measures the discrepancy in predictions between two probability distributions. When models, including several covariates (e.g., time interval in retention models), are formulated into a probability model, K-L divergence averaged over all possible values of model covariates, namely, "global" K-L divergence, could serve as a useful measure for model discrepancy. It is also reasonable to consider K-L divergence only averaged across those values of model covariates at which observations are taken in a specific experiment, the so-called "local" K-L divergence. In the two simulation studies, both the "global" K-L divergence and the "local" K-L divergence are relatively consistent with ADO-estimated posterior model probabilities. The fact that they are not entirely consistent is due to their mathematical nonequivalence and the asymptotic behavior of this consistency could be further examined by increasing binomial sample size n to a large number.

5.2 Discussion and Implications

When ADO is applied to empirical experiments for model discrimination, the data are generated by the cognitive mechanisms which may not be exactly the same with any existing models. Models often represent approximations to the true underlying situations. The problem of model misspecification is unavoidable because all possible candidate models that could be pre-specified for ADO to discriminate may be wrong for the true

data generating process. In practice, exactly identifying the data-generating model may not be possible. Often we discriminate models not to identify the exactly true model but to recognize the best approximate model. It has been verified that under the set-up of present simulations, ADO model discrimination would favor the candidate model whose prediction is most similar to the underlying process. Therefore, the present thesis provides simulation-based data that support the robustness of ADO when its assumption of one candidate model generating the data has been violated. In other words, ADO could still be used with confidence since it is likely to find the most similar model even if not the exactly correct model. The ADO algorithm for model discrimination is already applied in psychology (e.g., Myung & Pitt, 2009; Daniel R. Cavagnaro, Myung, Pitt, & Kujala, 2010; D. R. Cavagnaro, Pitt, & Myung, 2011), and the current results increase our confidence in further applying ADO to empirical experiments where the data-generating model is likely unknown.

5.3 Limitations

There are several limitations of the present thesis.

First, besides Kullback-Leibler divergence, there exist many other measures of model discrepancy. In the present thesis, if the model discrepancy between one model and the data-generating model is smaller compared to others, we consider this model a better approximation of the data-generating model. Since model discrepancy is an important concept in evaluating which model is a better approximation, relying on only one measure could be unwise. It could be useful in reducing the potential biasness if other

probability distance measures such as Kolmogorov-Smirnov distance are also taken into account.

Second, there is another limitation of expected K-L divergence. In measuring an HYP model and a POW model with fixed parameter values, we only need a single value of expected K-L divergence. However, in measuring an HYP model and a candidate model POW, which consists of predictions from a group of POW models with different parameter values, there is a distribution of expected K-L divergences instead of a single value, causing inconvenience in summarizing the results.

Third, the present results are obtained in limited simulation settings. Only retention models are included; however, there are many different models in other content areas such as decision making models or categorization models. Furthermore, the two simulations in the present thesis only include two scenarios of model misspecification, while many other situations of model misspecification may be of interest. For example, the data-generating model could be a *δ-contamination* model in which data are generated from one model with probability δ and another model with probability $1 - \delta$.

5.4 Future Directions

One natural extension of the current study is to consider ADO model discrimination under model misspecification for models in other content areas, such as decision-making models, categorization models, etc. It is of theoretical interest whether for different types of models, ADO model discrimination is still capable of selecting the candidate model that yields the most similar predictions to the data-generating model. The applicability of

expected K-L divergence in measuring model discrepancy also needs to be verified for different types of cognitive models.

Another topic of interest for future work is to explore ADO parameter estimation under model misspecification. In parameter estimation problems, a single candidate model including its functional form and parameter prior distribution has to be pre-specified. However, as discussed before, the underlying data generating process may likely be different from any candidate models. Under such circumstance of model misspecification, it is of theoretical interest to examine whether ADO for parameter estimation could still obtain an estimated model that is as similar as possible to the data-generating model in predictions.

It would be useful in exploring other frequently present model misspecification situations. For example, instead of data-generating model being a weighted mixture of POW and EXP, it could be formulated in such a way that with a certain probability the data are generated by POW, otherwise the data are generated by EXP. Another approach is to consider contamination models where the data are generated by POW or EXP with some extra random error.

An alternative to the problem of model misspecification is flexible model building. Such an option is provided by the recent advances in statistics, such as nonparametric function estimation through roughness penalty, which has already been developed for and applied in regression with Gaussian and non-Gaussian responses, hazard rate estimation, generalized linear model, and etc (Gu, 2002; Green & Silverman, 1994).

References

Allen, T. T., Liyang, Y., & Schmitz, J. (2003). An Experimental Design Criterion for Minimizing Meta-Model Prediction Errors Applied to Die Casting Process Design. *Journal of the Royal Statistical Society. Series C (Applied Statistics),* **52**, 103-117.

Amari, S.-i., & Nagaoka, H. (2000). *Methods of Information Geometry* (Vol. 191). OXFORD: University Press.

Amzal, B., Bois, F. Y., Parent, E., & Robert, C. P. (2006). Bayesian-Optimal Design via Interacting Particle Systems. *Journal of the American Statistical Association,* **101**, 773-785.

Atkinson, A. C., & Donev, A. N. (1992). Optimum experimental designs. England: Oxford University Press.

Bardsley, W. G., Wood, R. M. W., & Melikhova, E. M. (1996). Optimal Design: A Computer Program to Study the Best Possible Spacing of Design Points for Model Discrimination. *Computers Chem,* **20**, 145-157.

Borth, D. M. (1975). A Total Entropy Criterion for the Dual Problem of Model Discrimination and Parameter Estimation. *Journal of the Royal Statistical Society. Series B (Methodological),* **37**, 77-87.

Box, G. E. P., & Hill, W. J. (1967). Discrimination among Mechanistic Models. *Technometrics,* **9**, 57-71.

Box, G. E. P. (1979). Some problems of statistics and everyday life. Journal of the American Statistical Association, **74**, 1-4.

Box, M. J. (1969). Planning Experiments to Test the Adequacy of Non-Linear Models. *Journal of the Royal Statistical Society. Series C (Applied Statistics),* **18**, 241-248.

Casella, G., & Berger, R. L. (2001). *Statistical Inference* (2nd ed.): Duxbury Press.

Cavagnaro, D. R., Myung, J. I., Pitt, M. A., & Kujala, J. V. (2010). Adaptive Design Optimization: A Mutual Information-Based Approach to Model Discrimination in Cognitive Science. *Neural Computation,* **22**, 887-905.

Cavagnaro, D. R., Pitt, M. A., & Myung, J. I. (2011). Model discrimination through adaptive experimentation. *Psychon Bull Rev,* **18**, 204-210.

Chaloner, K., & Verdinelli, I. (1995). Bayesian experimental design: a review. *Statistical Science,* **10**, 273-304.

Cohen, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, **15**, 201-221.

Cover, T. M., & Thomas, J. A. (1991). Elements of information theory. New York: Wiley.

El-Gamal, M. A., & Palfrey, T. R. (1996). Economical Experiments: Bayesian Efficient Experiment Design. *International Journal of Game Theory*, **25**, 495-517.

Fan, W., & Hancock, G. R. (2011). Robust Means Modeling: An Alternative for Hypothesis Testing of Independent Means Under Variance Heterogeneity and Nonnormality. *Journal of Educational and Behavioral Statistics,* **37**, 137-156.

Farcomeni, A., & Ventura, L. (2012). An overview of robust methods in medical research. *Stat Methods Med Res,* **21**, 111-133.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis*. Boca Raton, FL: Champan and Hall/CRC.

Green, P. J., & Silverman, B. W. (1994). *Nonparametric regression and generalized linear models: a roughness penalty approach*. London: Chapman & Hall.

Grunwald, P. (2000). Model Selection Based on Minimum Description Length. *Journal Mathematical Psychology*, **44**, 133-152.

Gu, C. (2002). *Smoothing spline ANOVA models*: Springer.

Huber, P. J. (1981). *Robust Statistics*. New Yokr: Wiley.

Huber, P. J. (2002). John W. Tukey's contributions to robust statistics. *The Annals of Statistics,* **30**, 1640-1648.

Kujala, J., & Lukka, T. (2006). Bayesian adaptive estimation: The next dimension. *Journal of Mathematical Psychology*, **50**, 369-389.

Kujala, J. V. (2010). Obtaining the best value for money in adaptive sequential estimation. *Journal of Mathematical Psychology,* **54**, 475-480.

Kujala, J. V., Richardson, U., & Lyytinen, H. (2010). A Bayesian-optimal principle for learner-friendly adaptation in learning games. *Journal of Mathematical Psychology,* **54**, 247-255.

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *Annals of Mathematical Statistics* **22**, 79-86.

Lesmes, L. A., Jeon, S. T., Lu, Z. L., & Dosher, B. A. (2006). Bayesian adaptive estimation of threshold versus contrast external noise functions: the quick TvC method. *Vision Res,* **46**, 3160-3176.

Lesmes, L. A., Lu, Z. L., Baek, J., & Albright, T. D. (2010). Bayesian adaptive estimation of the contrast sensitivity function: the quick CSF method. [Research Support, N.I.H., Extramural]. *J Vis,* **10**, 17 11-21.

Lindley, D. V. (1956). On a measure of the information provided by an experiment. *The annals of mathematical statistics*, **27**, 986-1005.

Magis, D., & De Boeck, P. (2011). A Robust Outlier Approach to Prevent Type I Error Inflation in Differential Item Functioning. *Educational and Psychological Measurement,* **72**, 291-311.

Muller, P., Berry, D., Grieve, A., Smith, M., & Krams, M. (2007). Simulation-based sequential Bayesian design. *Journal of Statistical Planning and Inference,* **137**, 3140-3150.

Müller, P., Sansó, B., & De Iorio, M. (2004). Optimal Bayesian Design by Inhomogeneous Markov Chain Simulation. *Journal of the American Statistical Association,* **99**, 788-798.

Myung, J., Navarro, D., & Pitt, M. (2006). Model selection by normalized maximum likelihood. *Journal of Mathematical Psychology,* **50**, 167-179.

Myung, J. I., & Pitt, M. A. (2009). Optimal experimental design for model discrimination. [Research Support, N.I.H., Extramural]. *Psychological Review,* **116**, 499-518.

Navarro, D. J., Pitt, M. A., & Myung, I. J. (2004). Assessing the distinguishability of models and the informativeness of data. *Cogn Psychol,* **49**, 47-84.

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences,* **6**, 421-425.

Pitt, M. A., Myung, I. J., & Zhang, S. (2002). Toward a method of selecting among computational models of cognition. *Psychological Review,* **109**, 472-491.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: theory and data for two-choice decision tasks. *Neural Computation,* **20**, 873-922.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review,* **107**, 358-367.

Rubin, D. C., & Wenzel, A. E. (1996). One Hundred Years of Forgetting A Quantitative Description of Retention. *Psychological Review,* **103**, 734-760.

Schwaab, M., Silva, F., Queipo, C., Barretojr, A., Nele, M., & Pinto, J. (2006). A new approach for sequential experimental design for model discrimination. *Chemical Engineering Science,* **61**, 5791-5806.

Settles, B. (2010). Active Learning Literature Survey. *Computer Sciences Techinical Report 1648*.

Su, Y., Myung, I. J., & Pitt, M. A. (2005). Minimum description length and cognitive modeling. In I. J. Myung & M. A. Pitt (Eds.), *Advances in Minimum Description Length: Theory and Applications* (pp. 411-433): MIT Press.

Wagenmakers, E. J., Ratcliff, R., Gomez, P., & Iverson, G. J. (2004). Assessing model mimicry using the parametric bootstrap. *Journal of Mathematical Psychology,* **48**, 28-50.

Wainer, H. (1976). Robust Statistics: A Survey and Some Prescriptions. *Journal of Educational and Behavioral Statistics,* **1**, 285-312.

Yun, T., Young, C., Myung, J., Pitt, M., & Opfer, J. Optimal inference and feedback for representational change. Unpublished manuscript.

Zhang, S., & Lee, M. D. (2010). Optimal experimental design for a class of bandit problems. *Journal of Mathematical Psychology,* **54**, 499-508.

Appendix A: Tables and Figures

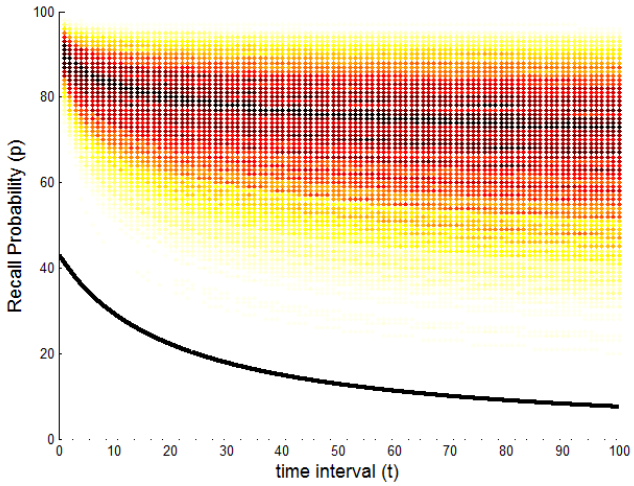
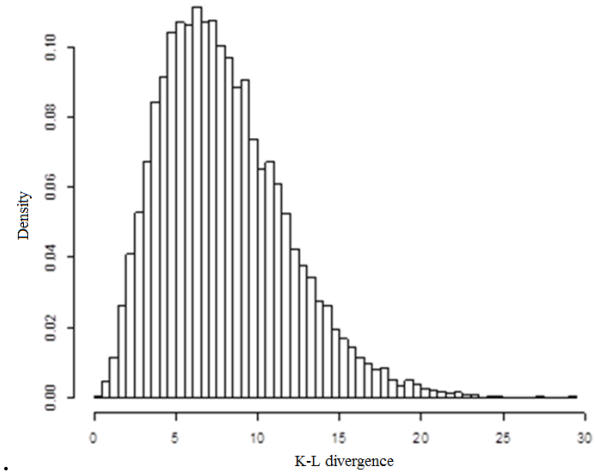


Figure 1: Heat map of candidate model POW

Each curve represents one single POW model with fixed parameter values; darker color indicates higher occurrences. 1 unit in Y axis represents 0.01 in probability of recalling. 0.01 is precision of the graph. The black curve represents the data-generating HYP1 model



.

Figure 2: Histogram of K-L divergence between HYP1 data-generating model and parametric family of POW models

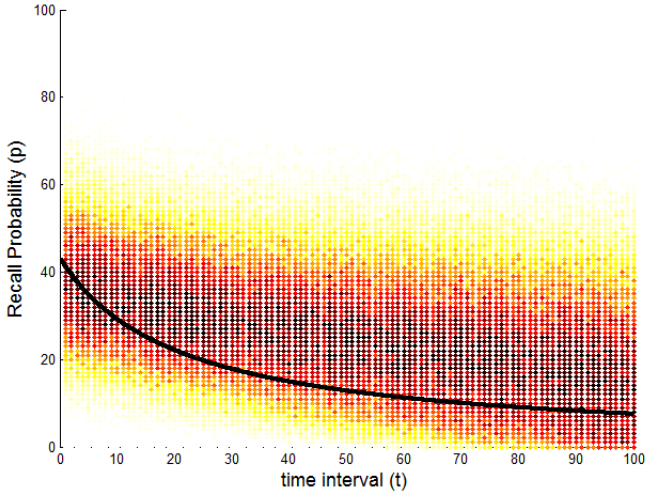


Figure 3: Heat map of candidate model EXP

Each curve represents one single EXP model with fixed parameter values; darker color indicates higher occurrences. 1 unit in Y axis represents 0.01 in probability of recalling. 0.01 is precision of the graph. The black curve represents the data-generating HYP1 model.

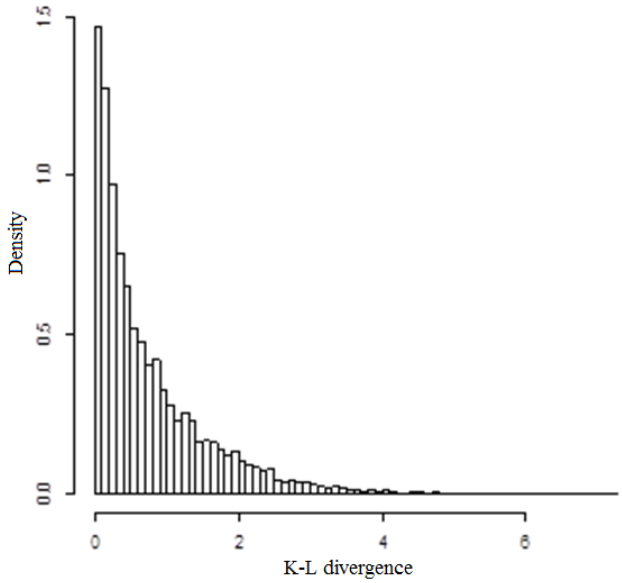


Figure 4: Histogram of K-L divergence between HYP1 data-generating model and parametric family of EXP models

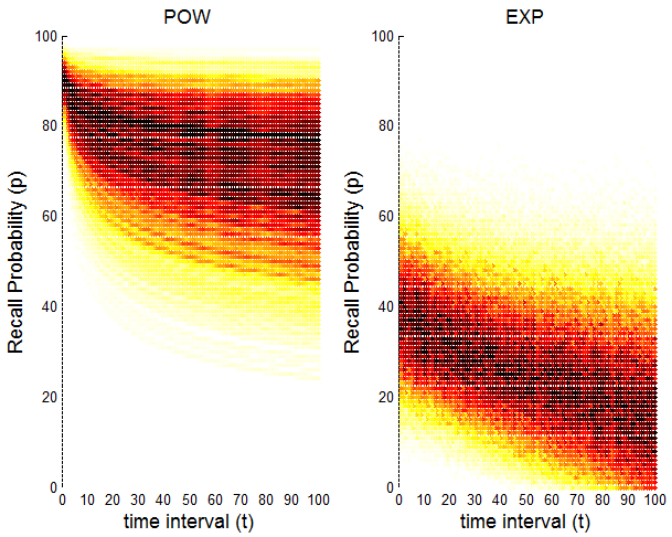


Figure 5: Heat maps of candidate model POW and EXP in high model discriminability condition

Each curve represents one single EXP or POW model with fixed parameter values; darker color indicates higher occurrences. 1 unit in Y axis represents 0.01 in probability of recalling. 0.01 is precision of the graph.

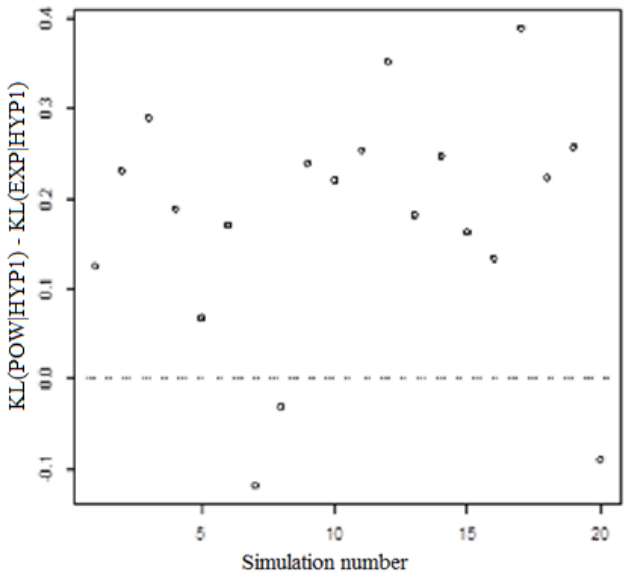


Figure 6: Comparison of K-L divergence between estimated POW and EXP with the data-generating model HYP1

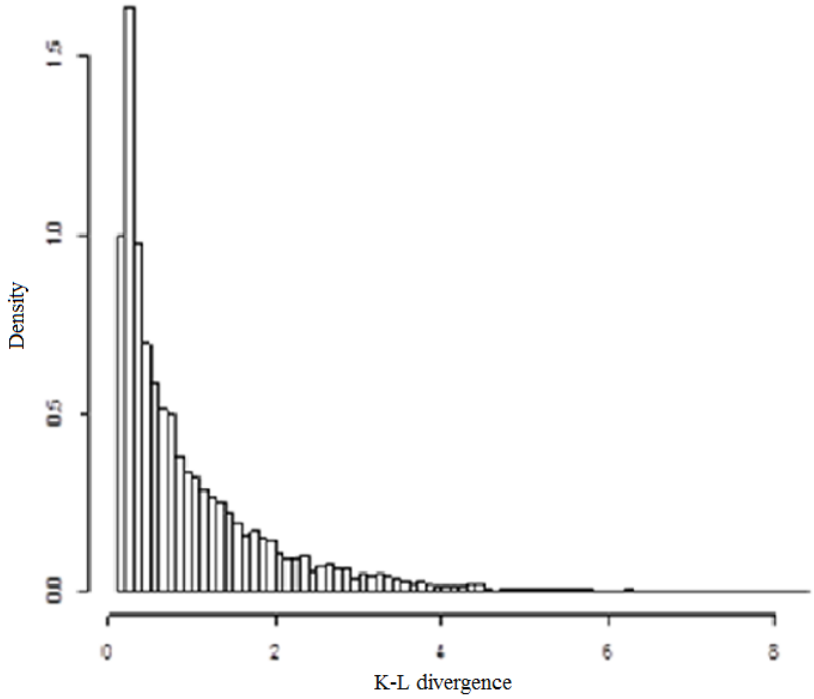


Figure 7: Histogram of K-L divergence between HYP2 data-generating model and candidate model POW

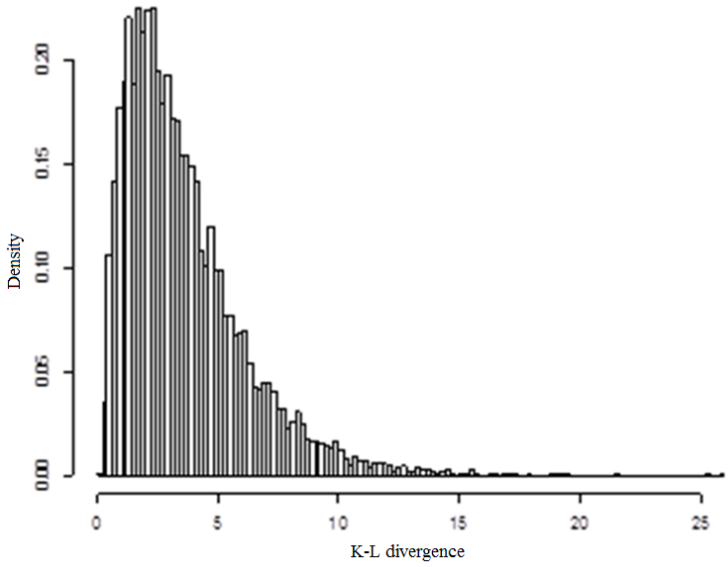


Figure 8: Histogram of K-L divergence between HYP2 data-generating model and candidate model EXP

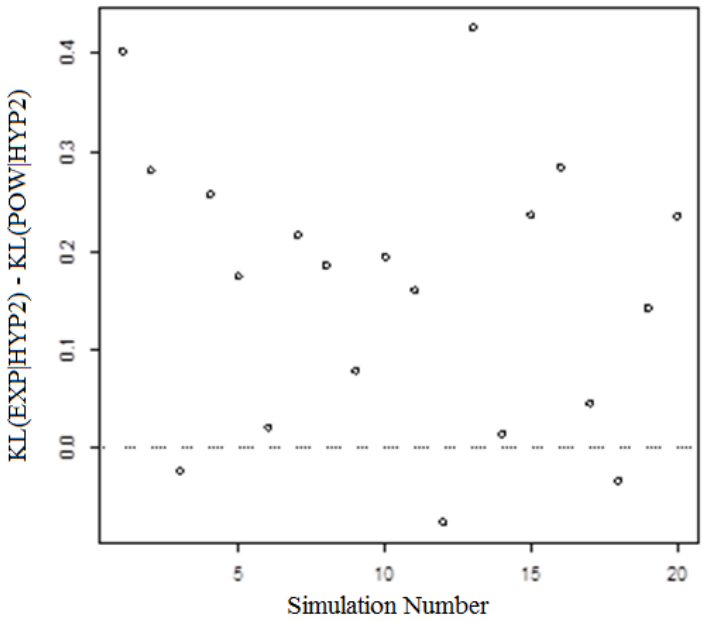


Figure 9: Comparison of K-L divergence between estimated EXP and data-generating model with POW with data-generating model HYP2

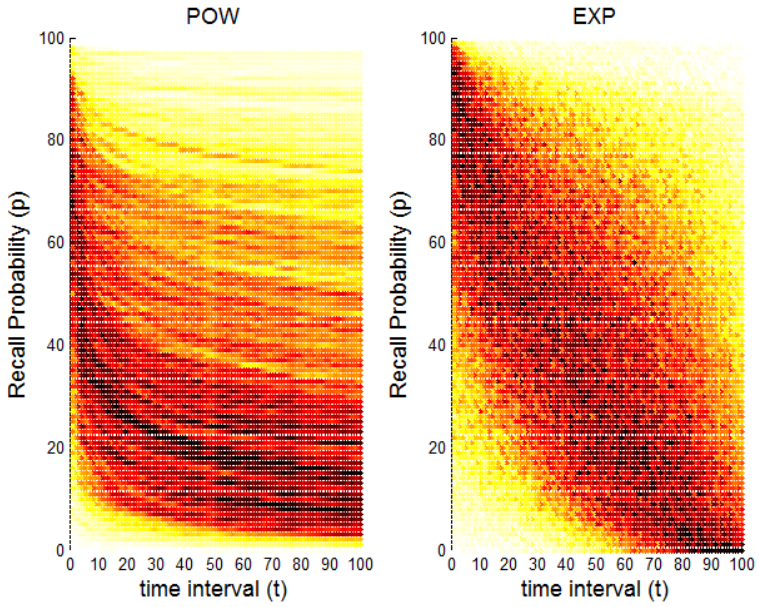


Figure 10: Heat maps of candidate model POW and EXP in low model discriminability condition

Each curve represents one single EXP or POW model with fixed parameter values; darker color indicates higher occurrences. 1 unit in Y axis represents 0.01 in probability of recalling. 0.01 is precision of the graph.

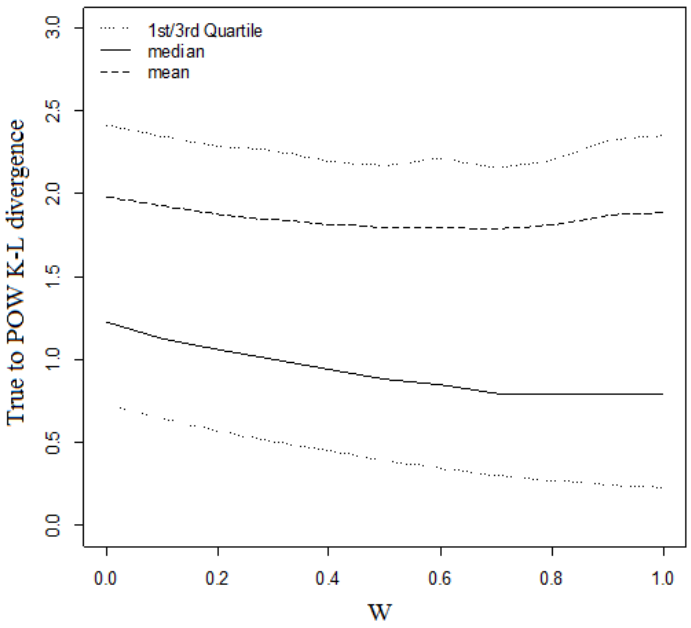


Figure 11: Summary of distributions of expected K-L divergence between different mixture data-generating models and parametric family of POW models

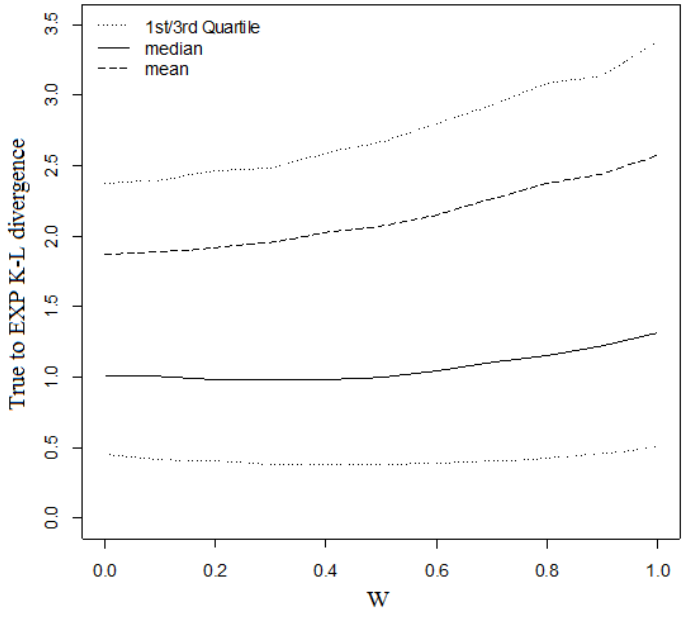


Figure 12: Summary of distributions of expected K-L divergence between different mixture data-generating models and parametric family of EXP models

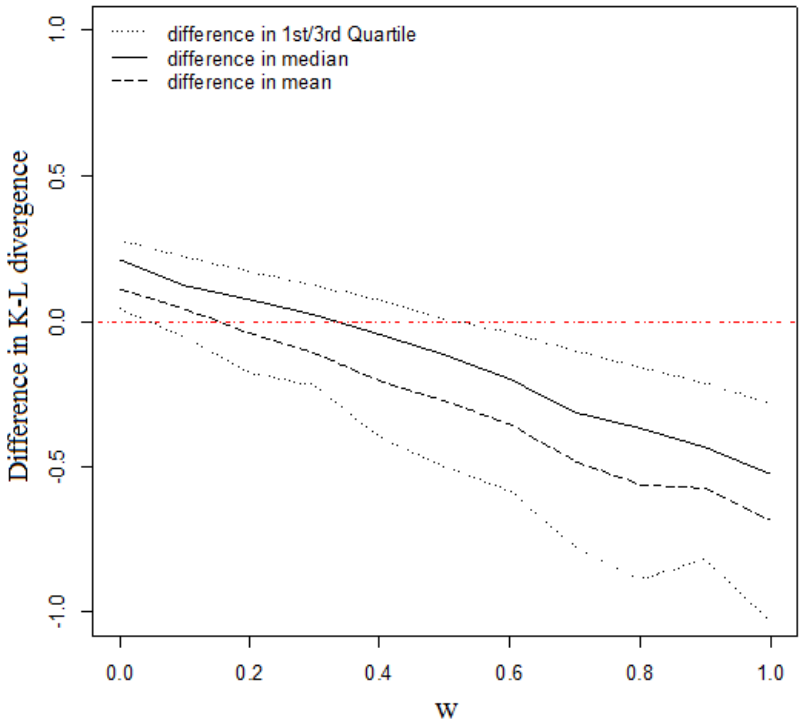


Figure 13: Difference in expected K-L divergence between different data-generating mixture models and candidate model POW and between different data-generating models and candidate model EXP

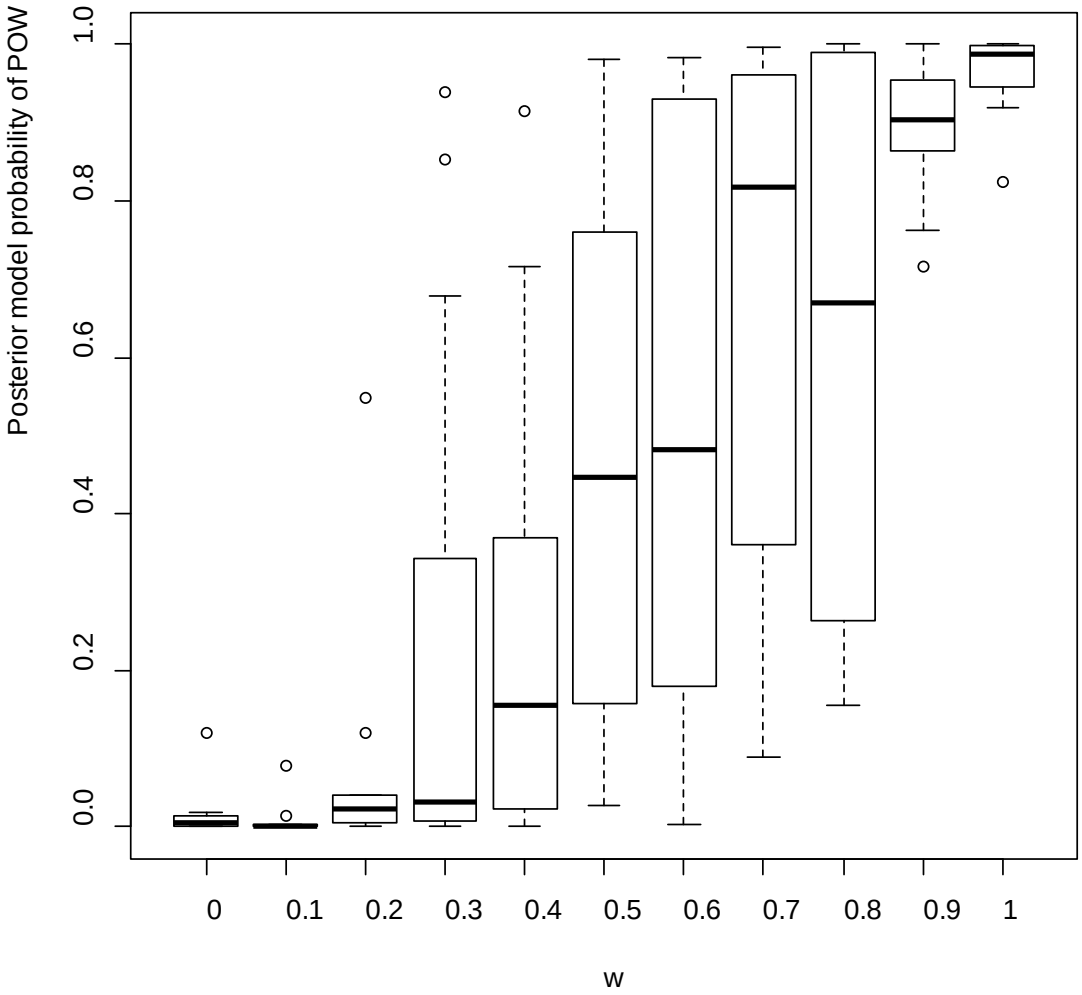


Figure 14: The posterior model probability of POW for different mixture data-generating models

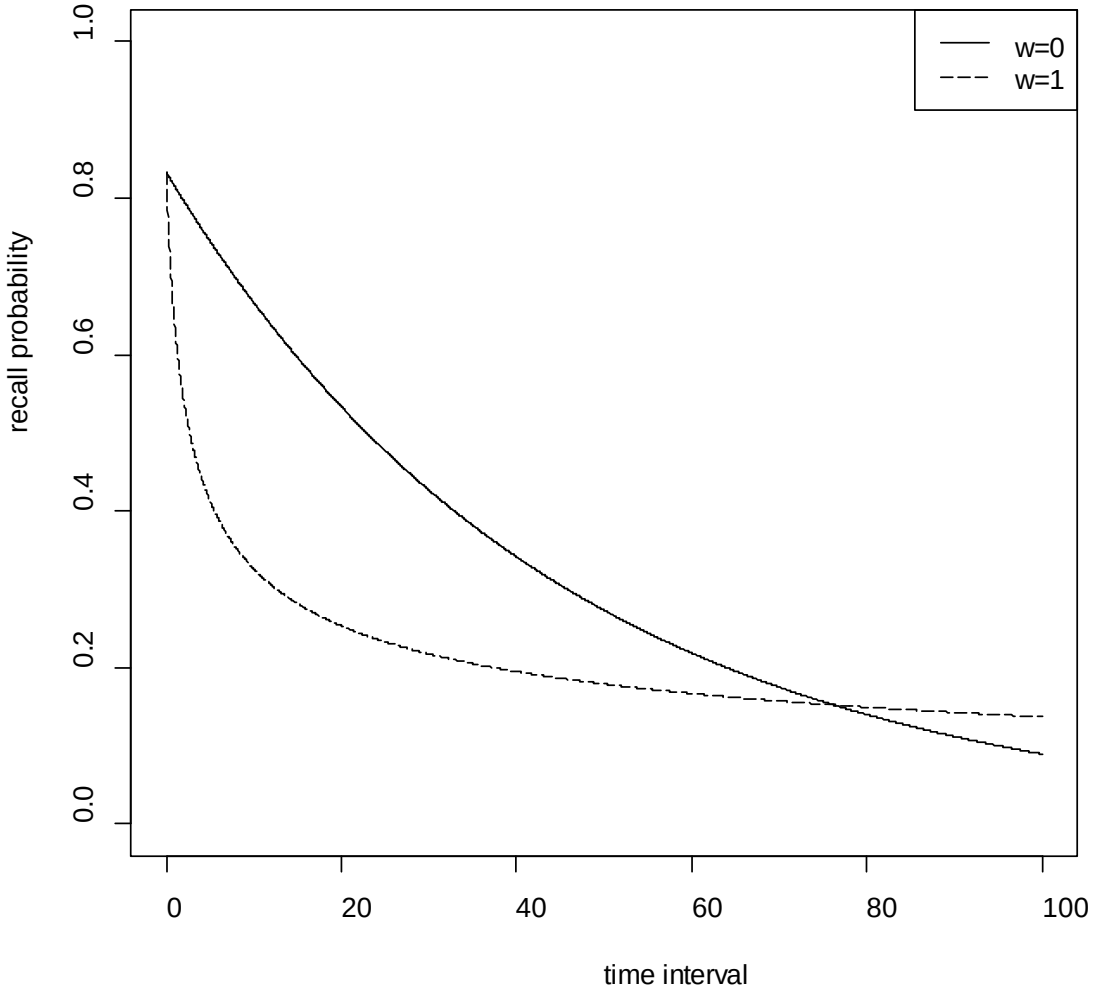


Figure 15: The mixture data-generating model when weight (w) equals 0 or 1

w	POW		EXP	
	a	b	a	b
0	0.8980±0.0302	0.2777±0.0411	0.7899±0.0862	0.0201±0.0062
0.1	0.9001±0.0233	0.2941±0.0596	0.7906±0.0701	0.0223±0.0070
0.2	0.8855±0.0404	0.2896±0.0259	0.7377±0.1038	0.0191±0.0039
0.3	0.8817±0.0241	0.3100±0.0442	0.7224±0.0565	0.0217±0.0039
0.4	0.8303±0.0822	0.2963±0.0485	0.6716±0.0930	0.0199±0.0067
0.5	0.7812±0.0765	0.2971±0.0558	0.6119±0.0734	0.0181±0.0039
0.6	0.8130±0.0788	0.3015±0.0534	0.6423±0.0823	0.0191±0.0064
0.7	0.7975±0.0555	0.3196±0.0507	0.5905±0.0602	0.0180±0.0044
0.8	0.7978±0.0832	0.3439±0.0621	0.5803±0.0779	0.0199±0.0059
0.9	0.7561±0.0629	0.3313±0.0721	0.5530±0.0697	0.0202±0.0069
1.0	0.8259±0.0940	0.3580±0.0746	0.5463±0.0854	0.0185±0.0068

Table 1: The parameter estimates (mean ± standard deviation) of model POW and EXP for different data-generating models

w	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	Total
Global K-L	1	0.9	0.9	0.8	0.65	0.6	0.45	0.5	0.65	1	1	0.712
Local K-L	1	1	0.9	0.8	0.8	0.7	0.7	0.75	0.65	1	1	0.803

Table 2: The proportion of consistency between three different K-L divergence based measure and posterior model probability results

Appendix B: C++ Code

Main programs for ADO simulation:

RepliDanMutu.cpp 87

Main programs for K-L divergence computation:

Priorkl.cpp 98

Other header files:

ClientFunctionsMutuADO.h 100

Distancefunction.h 104

RNGs.h 106

Randoma.h (not appended)

FastMatrix.h (not appended)

RepliDanMutu.cpp

```
//This is the main program of Adaptive Design Optimization, which maximizes the
//experiment utility function through sequential Monte Carlo methods.

// SequentualDO.cpp : main project file.

#include <fstream>
#include <iostream>
#include <time.h>
#include <math.h>
#include "randoma.h" //assembly version
using namespace std;

#include "FastMatrix.h"


#define USERTYPE double
typedef MATRIX<USERTYPE> MATRIXTYPE;
#define SN 10
#define N 1
#define THIN 200
#define BURN 5000
#define aPOWalpha 40
#define aPOWbeta 3
#define bPOWalpha 2
#define bPOWbeta 25
#define aEXPalpha 7
#define aEXPbeta 11
#define bEXPalpha 1
#define bEXPbeta 110
#define BFN 6000

#include "RNGs.h"
#include "ClientFunctionsMutuADO.h"
#include "distancefunction.h"

inline void SequentialDO(fstream& output) //PRIMARY ALGORITHM FUNCTION
{
	const int ASPT=5; //number of experiments for asymptotic evaluation

	const int NP = 30;		// number of particles
	const int NITER = 100;		// number of iterations
	const int T = 10;		// number of decision making points
	const int M=50000;
	const double Epsilon=0.00001;

	int iter=0, J, ip, j, i, Jp, Jd, t;

	USERTYPE u, uc, utilc, alpha, qratio=1, neff=0;
	MATRIXTYPE temp1(NP,1),temp2(1,N),temp3(NP,N);
	MATRIXTYPE
asptmeanvariance(ASPT,12),modelselection(ASPT,10),design(ASPT,10),datapost;
	MATRIXTYPE onesN(1,N),hundredsN(1,N), thousandsN(1,N),zerosN(1,N);
```

```
	MATRIXTYPE onesJ;
	MATRIXTYPE xxj(NP,N),xx(NP,N);
	MATRIXTYPE utiljp(1,NP),utilj(1,NP);
	MATRIXTYPE theta, parma, parmb, ppow,pexp;
	MATRIXTYPE xxc(1,N),xxf(1,N);
	MATRIXTYPE pobstemp(1,N),pobs(T,N);
	MATRIX<int> yobstemp(1,N),yobs(T,N), ypow,
yexp,yobstemppow,yobstempexp,temppow,tempexp;
	MATRIXTYPE xxt(T,N);
	MATRIXTYPE util(1,NP), accept(1,NP);
	MATRIXTYPE wgt(1,NP);
	MATRIXTYPE tempxxj,tempxxt;
	MATRIXTYPE wholeposteriorparmsPOW(M,2),wholeposteriorparmsEXP(M,2);
	int index;
	USERTYPE tempu, tempuc,tempbf_pow,tempbf_exp;
	USERTYPE apow,bpow,aexp,bexp,powat,powbt,expat,expbt,w;


	USERTYPE BF[T],model_prop_pow,model_prop_exp;

	w=0; //define the mixture model! 1: pow;   0: exp

	for (int aspt=0; aspt<ASPT; aspt++)
{
	cout<<"********"<<endl;
	cout<<"aspt="<<(aspt+1)<<endl;
	cout<<"********"<<endl;

	model_prop_pow=0.5; model_prop_exp=0.5;


	qratio=1;
	neff=0;
	datapost.reset(M,4);

	MATRIX<int> mno(1,NP);
	temp1=1;temp3.Random(&RandomDouble,1);
	temp2[0][0]=5;
	onesN=1;hundredsN=100;thousandsN=1000;zerosN=0;

	xxt=0;yobs=0;


	t=0;
	while(t<T)
	{
		cout <<'\n';
		cout << (t+1) << " *** ";


		xxj.InsertMatrix(temp1*temp2+(temp3*2-1)*0.05,0,0);



	if (t>0)
```

```
	{
		wholeposteriorparmsPOW=POW_postsampler(xxt,yobs,t-1,M,THIN);
		wholeposteriorparmsEXP=EXP_postsampler(xxt,yobs,t-1,M,THIN);
	}


	iter=0;
	Jp=20;
	theta.reset(Jp,2);parma.reset(Jp,N);parmb.reset(Jp,N);
	ppow.reset(Jp,N);pexp.reset(Jp,N);
	ypow.reset(Jp,N); yexp.reset(Jp,N);
	onesJ.reset(Jp,1);onesJ=1;

	//%----- Initial design samples
		ip=0;
		while(ip<NP)
		{
			tempxxt.reset(T,N); tempxxt=xxt;
			tempxxj=xxj.GetRow(ip);
			tempxxt.InsertMatrix(tempxxj,t,0);
			for(int ii=0; ii<Jp; ii++)
			{
			  if (t==0)
			  {

theta.InsertMatrix(BetaPrior(aPOWalpha,aPOWbeta,bPOWalpha,bPOWbeta),ii,0);
			  }
			  else
			  {
				index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsPOW.SubSet(index,index+1,0,2),ii,0);
			  }
			}
			parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
			ppow=parma.MultMembs((onesJ*tempxxj+1).Pow(-parmb));
			ypow=binornd(SN,ppow);

			for(int ii=0; ii<Jp; ii++)
			{
			   if (t==0)
			  {

theta.InsertMatrix(BetaPrior(aEXPalpha,aEXPbeta,bEXPalpha,bEXPbeta),ii,0);
			  }
			   else
			   {
				index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsEXP.SubSet(index,index+1,0,2),ii,0);
			   }

		};
			parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
```

```
                    pexp=parma.MultMembs(((-
parmb).MultMembs(onesJ*tempxxj)).Exp());
                    yexp=binornd(SN,pexp);

                    u=0;j=0;tempu=0;
                    while (j<Jp)
                    {
                           //plug in the proposed observations
                           temppow.reset(1,N);
                           tempexp.reset(1,N);
                           temppow=ypow.GetRow(j);
                           tempexp=yexp.GetRow(j);
                           yobstemppow.reset(T,N);
                           yobstempexp.reset(T,N);
                           yobstemppow=yobs;
                           yobstempexp=yobs;
                           yobstemppow.InsertMatrix(temppow,t,0);
                           yobstempexp.InsertMatrix(tempexp,t,0);

                           tempbf_pow=BF_PowoverExp(tempxxt,yobstemppow,t,BFN);
                           tempbf_exp=BF_EXPoverPow(tempxxt,yobstempexp,t,BFN);

                           tempu=model_prop_pow*log((tempbf_pow+1e-
300)/(tempbf_pow*model_prop_pow+1-model_prop_pow+1e-300)) + model_prop_exp *
log((tempbf_exp+1e-300)/(tempbf_exp*model_prop_exp+1-model_prop_exp+1e-300))+2.0;


                           if(tempu<Epsilon)
                           {
                                  u= u + log(Epsilon);

                           }
                           else
                           {
                                  u= u + log(tempu);
                           }
                           j++;
                    };
                    utiljp[0][ip]=exp(u);
                    ip++;
             }; //ip


             xxc=pickbest(utiljp,xxj,1);

             iter=iter+1;

      //++++ MCMC/SA ITERATION ++++++++++++++++++++++++++++++++++++/////////

      while(iter<NITER)
      {
             if ((iter+1)%1==0)
                    cout << iter << " - ";
```

```
            J=Round(2*log(Truncate(double(iter)/3)+1.0))+20; //% anneaing
schedule, Temp=1/J


          Jd=J-Jp;

          //% if then else ------------------
if (Jd > 0)
 {
           theta.reset(Jd,2);parma.reset(Jd,N);parmb.reset(Jd,N);
              ppow.reset(Jd,N);pexp.reset(Jd,N);
              ypow.reset(Jd,N); yexp.reset(Jd,N);
           onesJ.reset(Jd,1);onesJ=1;


              //%----- additional samples given a design
       ip=0;
              while(ip<NP)
              {
                     tempxxt.reset(T,N); tempxxt=xxt;
                     tempxxj=xxj.GetRow(ip);
                     tempxxt.InsertMatrix(tempxxj,t,0);

                     for(int ii=0; ii<Jd; ii++)
                     {
                       if (t==0)
                       {

theta.InsertMatrix(BetaPrior(aPOWalpha,aPOWbeta,bPOWalpha,bPOWbeta),ii,0);
                       }
                       else
                       {
                              index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsPOW.SubSet(index,index+1,0,2),ii,0);
                       }
            };

                     parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
                     ppow=parma.MultMembs((onesJ*tempxxj+1).Pow(-parmb));
                     ypow=binornd(SN,ppow);

                     for(int ii=0; ii<Jd; ii++)
                     {
                       if (t==0)
                       {

theta.InsertMatrix(BetaPrior(aEXPalpha,aEXPbeta,bEXPalpha,bEXPbeta),ii,0);
                       }
                       else
                       {
                              index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsEXP.SubSet(index,index+1,0,2),ii,0);
                       }
```

```
            };

                    parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
                    pexp=parma.MultMembs(((-
parmb).MultMembs(onesJ*tempxxj)).Exp());
                    yexp=binornd(SN,pexp);


                    u=0;j=0;tempu=0;
                    while (j<Jd)
                    {
                           temppow.reset(1,N);
                           tempexp.reset(1,N);
                           temppow=ypow.GetRow(j);
                           tempexp=yexp.GetRow(j);
                           yobstemppow.reset(T,N);
                           yobstempexp.reset(T,N);
                           yobstemppow=yobs;
                           yobstempexp=yobs;
                           yobstemppow.InsertMatrix(temppow,t,0);
                           yobstempexp.InsertMatrix(tempexp,t,0);

                           tempbf_pow=BF_PowoverExp(tempxxt,yobstemppow,t,BFN);
                           tempbf_exp=BF_EXPoverPow(tempxxt,yobstempexp,t,BFN);

                           tempu=model_prop_pow*log((tempbf_pow+1e-
300)/(tempbf_pow*model_prop_pow+1-model_prop_pow+1e-300)) + model_prop_exp *
log((tempbf_exp+1e-300)/(tempbf_exp*model_prop_exp+1-model_prop_exp+1e-300))+2.0;


                           if(tempu<Epsilon)
                           {
                                  u=u + log(Epsilon);

                           }
                           else
                           {
                                  u=u + log(tempu);
                           }
                           j++;
                    };
                    utilj[0][ip]=exp(u);
                    ip++;
             }; //ip



             //%----- resampling step----------------------------------------------
-
             wgt=utilj/(utilj.Sum());

             neff=1./(wgt.MultMembs(wgt).Sum());
             if (neff < float(NP)*0.95)
                 {mno=multinorm(NP,wgt);}
```

```
            else
            {mno=1;}

            util=0;xx=0;ip=0;j=0;
            while(ip<NP)
            {
                   i=0;
                   while (i<mno[0][ip])
                   {
                          util[0][j]=utiljp[0][ip]*utilj[0][ip];
                          xx.InsertMatrix(xxj.GetRow(ip), j, 0);
                          j++;
                          i++;
                   };
                   ip++;
            };
   }
   else
   {
            util=utiljp;
            xx=xxj;
   };

         //%----- MH step ----------------------------------------
            ip=0;accept=0;
            theta.reset(J,2);parma.reset(J,N);parmb.reset(J,N);
            ppow.reset(J,N);pexp.reset(J,N);
            ypow.reset(J,N); yexp.reset(J,N);
            onesJ.reset(J,1);onesJ=1;
            while (ip < NP)
            {
                   tempxxt.reset(T,N); tempxxt=xxt;

                   xxc=xx.GetRow(ip).MultMembs(onesN+randn(N)*.9);// 0.9
                   //xxc=xx.GetRow(ip) + randn(N)*5;

                   tempxxt.InsertMatrix(xxc,t,0);

                   if (((xxc > zerosN).Sum() < N) || ((hundredsN > xxc).Sum() <
N))
                          {
                            qratio=0.0;
                          }
                   else
                          {
                                 //qratio=1.0;

                                 qratio=0.0;
                                 for(int ii=0; ii<N; ii++)
                                 {
                                 u=pow((xx[ip][ii]-
xxc[0][ii]),2)*(pow(xx[ip][ii],2)-
pow(xxc[0][ii],2))/(2.0*0.81*pow(xx[ip][ii],2)*pow(xxc[0][ii],2));
                                    qratio=qratio+log(xx[ip][ii])-log(xxc[0][ii])-u;
                                    }
```

```
                              qratio=exp(qratio);

                        };

                  for(int ii=0; ii<J; ii++)
                  {
                        if (t==0)
                    {

theta.InsertMatrix(BetaPrior(aPOWalpha,aPOWbeta,bPOWalpha,bPOWbeta),ii,0);
                    }
                        else
                    {
                          index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsPOW.SubSet(index,index+1,0,2),ii,0);
                    }
            };

                  parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
                  ppow=parma.MultMembs((onesJ*xxc+1).Pow(-parmb));
                  ypow=binornd(SN,ppow);

                  for(int ii=0; ii<J; ii++)
                  {
                     if (t==0)
                    {

theta.InsertMatrix(BetaPrior(aEXPalpha,aEXPbeta,bEXPalpha,bEXPbeta),ii,0);
                    }
                     else
                    {
                          index=TIRandom(0,M-1);

theta.InsertMatrix(wholeposteriorparmsEXP.SubSet(index,index+1,0,2),ii,0);
                    }
            };

                  parma=theta.GetColumn(0)*onesN;parmb=theta.GetColumn(1)*onesN;
                  pexp=parma.MultMembs((-parmb.MultMembs(onesJ*xxc)).Exp());
                  yexp=binornd(SN,pexp);

                  uc=0;j=0;tempuc=0;
                  while (j<J)
                  {
                        temppow=ypow.GetRow(j);
                        tempexp=yexp.GetRow(j);
                        yobstemppow.reset(T,N);
                        yobstempexp.reset(T,N);
                        yobstemppow=yobs;
                        yobstempexp=yobs;
                        yobstemppow.InsertMatrix(temppow,t,0);
                        yobstempexp.InsertMatrix(tempexp,t,0);

                        tempbf_pow=BF_PowoverExp(tempxxt,yobstemppow,t,BFN);
```

```
                        tempbf_exp=BF_EXPoverPow(tempxxt,yobstempexp,t,BFN);

                        tempuc=model_prop_pow*log((tempbf_pow+1e-
300)/(tempbf_pow*model_prop_pow+1-model_prop_pow+1e-300)) + model_prop_exp *
log((tempbf_exp+1e-300)/(tempbf_exp*model_prop_exp+1-model_prop_exp+1e-300))+2.0;


                        if(tempuc<Epsilon)
                        {
                              uc=uc + log(Epsilon);

                        }
                        else
                        {
                              uc=uc + log(tempuc);
                        }
                        j++;
                  };
                  utilc=exp(uc);
      //new

                  alpha=(qratio*utilc)/util[0][ip];
                  if (alpha>1)
                  {
                        alpha=1;
                  };    //% acceptance probability XXXXXXXXXXXXXXXXX
                  if (RandomDouble() < alpha)
                  {
                        xx.InsertMatrix(xxc,ip,0);
                        util[0][ip]=utilc;
                        accept[0][ip]=1;
                  } //% Metropolis-Hastings algorithm
                        //%----- record the results
                  ip++;
            }

            xxf=pickbest(util,xx,1);
            xxj=xx;Jp=J;utiljp=util;
            iter++;
            };

   //%----- Update Posteriors
      //use the last xxf to generate "observed data"
      xxt.InsertMatrix(xxf,t,0);

      pobstemp=((xxf+1).Pow(-.391))*.833*w+((xxf*(-.0223)).Exp())*.833*(1-w);

      yobstemp=binornd(SN,pobstemp);
      yobs.InsertMatrix(yobstemp,t,0);

      BF[t]=BF_EXPoverPow(xxt,yobs,t,50000);
      model_prop_exp=BF[t]/(1+BF[t]);
      model_prop_pow=1-model_prop_exp;

      modelselection[aspt][t]=model_prop_pow;
```

```
	if(t==T-1)
	{
		datapost.InsertMatrix(POW_postsampler(xxt,yobs,t,M,THIN),0,0);
		datapost.InsertMatrix(EXP_postsampler(xxt,yobs,t,M,THIN),0,2);
	}

	t++;

	};

	design.InsertMatrix(xxt.TransPose(),aspt,0);

	asptmeanvariance.InsertMatrix(datapost.MeanByColumn(),aspt,0);
	asptmeanvariance[aspt][4]=(datapost.GetColumn(0)).VarAll();
	asptmeanvariance[aspt][5]=(datapost.GetColumn(1)).VarAll();
	asptmeanvariance[aspt][6]=(datapost.GetColumn(2)).VarAll();
	asptmeanvariance[aspt][7]=(datapost.GetColumn(3)).VarAll();

	apow=asptmeanvariance[aspt][0];
	bpow=asptmeanvariance[aspt][1];
	aexp=asptmeanvariance[aspt][2];
	bexp=asptmeanvariance[aspt][3];

	powat=0.9;
	powbt=0.08;
	expat=0.34;
	expbt=0.01;

	asptmeanvariance[aspt][8]=overtKLPQ(ProbMix,ProbPow,powat,powbt,expat,expbt
,w,apow,bpow);
	asptmeanvariance[aspt][9]=overtKLPQ(ProbMix,ProbExp,powat,powbt,expat,expbt
,w,aexp,bexp);
	asptmeanvariance[aspt][10]=localKLPQ(ProbMix,ProbPow,powat,powbt,expat,expb
t,w,apow,bpow,xxt);
	asptmeanvariance[aspt][11]=localKLPQ(ProbMix,ProbExp,powat,powbt,expat,expb
t,w,aexp,bexp,xxt);



	}
	asptmeanvariance.Export(output);output << '\n';
	design.Export(output); output<<'\n';
	modelselection.Export(output); output<<'\n';
};

int main(int argc, char* argv[]) //ENTRYPOINT FOR PROGRAM, PERFORM TESTING HERE
{
	fstream output;


	#ifdef DEBUG
	try{
	#endif
```

```
	TRandomInit(int(time(NULL)));
	//srand(int(time(NULL)));

	output.open("SeqDOoutput.txt", fstream::out); //initialize output stream
    SequentialDO(output);output << '\n';output.close();

	#ifdef DEBUG
	}
	catch(char* e)
	{
		output << e;
		cerr << e;
	}
	#endif

	return 0;
}
```

Priorkl.cpp

```
//This program estimates the Kullback-Leibler divergence between the data-
//generating model and the candidate models.

#include <fstream>
#include <iostream>
#include <time.h>
#include <math.h>
#include "randoma.h" //assembly version
using namespace std;

#include "FastMatrix.h"


#define USERTYPE double
typedef MATRIX<USERTYPE> MATRIXTYPE;
#define SN 10
#define N 1
#define THIN 200
#define BURN 5000
#define aPOWalpha 40
#define aPOWbeta 3
#define bPOWalpha 2
#define bPOWbeta 25
#define aEXPalpha 7
#define aEXPbeta 11
#define bEXPalpha 1
#define bEXPbeta 110
#define BFN 6000

#include "RNGs.h"
#include "ClientFunctionsMutuADO.h"
#include "distancefunction.h"

inline void SequentialDO(fstream& output) //PRIMARY ALGORITHM FUNCTION
{
       const int INTER=20000;
       double apow,bpow,aexp,bexp,powat,powbt,expat,expbt,w;
       MATRIXTYPE answer(INTER,10);
       MATRIXTYPE theta;
       double inter=0;

       while(inter<INTER)
       {
              cout<<(inter+1)<<"--";
        w=0;

              powat=0.90;
              powbt=0.08;
              expat=0.34;
              expbt=0.01;
              answer[inter][0]=powat;
              answer[inter][1]=powbt;
```

```
		answer[inter][2]=expat;
		answer[inter][3]=expbt;


		theta=BetaPrior(aPOWalpha, aPOWbeta, bPOWalpha, bPOWbeta);
		apow=theta[0][0];
		bpow=theta[0][1];
		answer.InsertMatrix(theta,inter,4);

	answer[inter][6]=overtKLPQ(ProbMix,ProbPow,powat,powbt,expat,expbt,w,apow,b
pow);

		theta=BetaPrior(aEXPalpha, aEXPbeta, bEXPalpha, bEXPbeta);
		aexp=theta[0][0];
		bexp=theta[0][1];
		answer.InsertMatrix(theta,inter,7);

	answer[inter][9]=overtKLPQ(ProbMix,ProbExp,powat,powbt,expat,expbt,w,aexp,b
exp);


		inter++;
	}
	answer.Export(output);
};

int main(int argc, char* argv[]) //ENTRYPOINT FOR PROGRAM, PERFORM TESTING HERE
{
	fstream output;


	#ifdef DEBUG
	try{
	#endif

	TRandomInit(int(time(NULL)));
	//srand(int(time(NULL)));

	   output.open("SeqDOoutput.txt", fstream::out); //initialize output stream
    SequentialDO(output);output << '\n';output.close();

	   #ifdef DEBUG
	   }
	   catch(char* e)
	   {
		output << e;
		cerr << e;
	   }
	   #endif

	   return 0;
}
```

ClientFunctionsMutuADO.h

```
//This section includes several functions such as calculating Bayes factors,
//posterior sampling via a hybrid of Metropolis-Hasting and Gibbs sampler.

inline MATRIXTYPE BetaPrior(int aalpha, int abeta, int balpha, int bbeta)
{
	MATRIXTYPE y(1,2);
	y.InsertMatrix(RandBeta(aalpha, abeta,1),0,0);
	y.InsertMatrix(RandBeta(balpha, bbeta,1),0,1);
	return y;
}


inline double EXP_likifunc(USERTYPE parma, USERTYPE parmb, MATRIXTYPE &x,
MATRIX<int> &yobs, int t)
{
	USERTYPE u,v;
	MATRIXTYPE pexp;
	int i,j;
	pexp=(x*(-parmb)).Exp()*parma;
	j=t;u=0; //because t starts from zero in the main program
	while(j>=0)
	{
		i=0;v=0;
		while(i<N)
		{
			v=v+log(pow(pexp[j][i],yobs[j][i])*pow((1-pexp[j][i]),(SN-
yobs[j][i])));
			i++;
		}
		u=u+v;
		j--;
	}
	u=exp(u);
	return u;
}

inline double POW_likifunc(USERTYPE parma, USERTYPE parmb, MATRIXTYPE &x,
MATRIX<int> &yobs, int t)
{
	USERTYPE u,v;
	MATRIXTYPE ppow;
	int i,j;
	ppow=(x+1).Pow(-parmb)*parma;
	j=t;u=0; //because t starts from zero in the main program
	while(j>=0)
	{
		i=0;v=0;
		while(i<N)
		{
			v=v+log(pow(ppow[j][i],yobs[j][i])*pow((1-ppow[j][i]),(SN-
yobs[j][i])));
			i++;
```

```
		}
		u=u+v;
		j--;
	}
	u=exp(u);
	return u;
}


inline MATRIXTYPE EXP_postsampler(MATRIXTYPE &x, MATRIX<int> &yobs, int t, int n,
int TH) //Independent G/M-H Algorithm
{
	MATRIXTYPE y,r;
	y.reset(n+BURN,2);
	double a_cur, b_cur, a_prop, b_prop;
	double cur,prop;
	int i,j;
	i=0;
	a_cur=RandomDouble();
	b_cur=RandomDouble();
	while(i<(n+BURN))
	{
		j=0;
		while(j<TH)
		{
			a_prop=RandomDouble();
			cur=EXP_likifunc(a_cur,b_cur,x,yobs,t)*pow(a_cur,(aEXPalpha-
1))*pow((1-a_cur),(aEXPbeta-1));

	prop=EXP_likifunc(a_prop,b_cur,x,yobs,t)*pow(a_prop,(aEXPalpha-1))*pow((1-
a_prop),(aEXPbeta-1));
			if(RandomDouble() < prop/cur)
			{
				a_cur=a_prop;
			}
			b_prop=RandomDouble();
			cur=EXP_likifunc(a_cur,b_cur,x,yobs,t)*pow(b_cur,(bEXPalpha-
1))*pow((1-b_cur),(bEXPbeta-1));

	prop=EXP_likifunc(a_cur,b_prop,x,yobs,t)*pow(b_prop,(bEXPalpha-1))*pow((1-
b_prop),(bEXPbeta-1));
			if(RandomDouble() < prop/cur)
			{
				b_cur=b_prop;
			}
			j++;
		}
		y[i][0]=a_cur;
		y[i][1]=b_cur;
		i++;
	}
	r=y.SubSet(BURN,n+BURN,0,2);
	return r;
}
```

```
inline MATRIXTYPE POW_postsampler(MATRIXTYPE &x, MATRIX<int> &yobs, int t, int n,
int TH)//Independent G/M-H Algorithm
{
	MATRIXTYPE y,r;
	y.reset(n+BURN,2);
	double a_cur, b_cur, a_prop, b_prop;
	double cur,prop;
	int i,j;
	i=0;
	a_cur=RandomDouble();
	b_cur=RandomDouble();
	while(i<(n+BURN))
	{
		j=0;
		while(j<TH)
		{
			a_prop=RandomDouble();
			cur=POW_likifunc(a_cur,b_cur,x,yobs,t)*pow(a_cur,(aPOWalpha-
1))*pow((1-a_cur),(aPOWbeta-1));

	prop=POW_likifunc(a_prop,b_cur,x,yobs,t)*pow(a_prop,(aPOWalpha-1))*pow((1-
a_prop),(aPOWbeta-1));
			if(RandomDouble() < prop/cur)
			{
				a_cur=a_prop;
			}
			b_prop=RandomDouble();
			cur=POW_likifunc(a_cur,b_cur,x,yobs,t)*pow(b_cur,(bPOWalpha-
1))*pow((1-b_cur),(bPOWbeta-1));

	prop=POW_likifunc(a_cur,b_prop,x,yobs,t)*pow(b_prop,(bPOWalpha-1))*pow((1-
b_prop),(bPOWbeta-1));
			if(RandomDouble() < prop/cur)
			{
				b_cur=b_prop;
			}
			j++;
		}
		y[i][0]=a_cur;
		y[i][1]=b_cur;
		i++;
	}
	r=y.SubSet(BURN,n+BURN,0,2);
	return r;
}

inline double BF_EXPoverPow(MATRIXTYPE &x,MATRIX<int> &yobs,int t,int n)
{
	USERTYPE mpow,mexp,BF;
	USERTYPE a,b,f;
	MATRIXTYPE theta;
	int i;
	i=0; mpow=0;
	while (i<n)
	{
```

```
		theta=BetaPrior(aPOWalpha,aPOWbeta,bPOWalpha,bPOWbeta);
		a=theta[0][0];
		b=theta[0][1];
		f=POW_likifunc(a,b,x,yobs,t);
		mpow=mpow+f;
		i++;
	}
	i=0; mexp=0;
	while (i<n)
	{
		theta=BetaPrior(aEXPalpha,aEXPbeta,bEXPalpha,bEXPbeta);
		a=theta[0][0];
		b=theta[0][1];
		f=EXP_likifunc(a,b,x,yobs,t);
		mexp=mexp+f;
		i++;
	}
	BF=mexp/mpow;
	return BF;
}


inline MATRIXTYPE pickbest(MATRIXTYPE &utility, MATRIXTYPE &design,int K)
{
	int W=design.GetWidth();
	int H=design.GetHeight();
	MATRIXTYPE augmat(H,W+1);
	MATRIXTYPE firstK(K,W);
	MATRIXTYPE ans(1,W);
	for(int i=0;i<H;i++)
	{
		augmat.InsertMatrix(utility.SubSet(0,1,i,i+1),i,0);
		augmat.InsertMatrix(design.SubSet(i,i+1,0,W),i,1);
	}
	augmat.SortRowsByCol(0);
	firstK=augmat.SubSet(H-K,H,1,W+1);
	firstK.SortSelfByRow();
	ans=firstK.MeanByColumn();
	return(ans);
}
```

Distancefunction.h

```
//This program estimates the Kullback-Leibler divergence between exponential,
//power, hyperbolic, or a convex combination of exponential and power model.

inline double ProbExp(double a,double b, double t)
{
	double y;
	y=a*exp(-b*t);
	return y;
}


inline double ProbPow(double a, double b, double t)
{
	double y;
	y=a*pow((t+1),(-b));
	return y;
}

inline double ProbHyperbolic(double a, double b, double t)
{
	double y;
	y=1/(a+b*t);
	return y;
}

inline double ProbMix(double powa,double powb, double expa, double expb, double w,
double t)
{
	double y;
	y=powa*pow((t+1),(-powb))*w + expa*exp(-expb*t)*(1-w);
	return y;
}

inline double PdfExp(double a, double b, double t, int yobs)
{
	double p;
	double y;
	p=ProbExp(a,b,t);
	y=pow(p,yobs)*pow((1-p),(SN-yobs));
	return y;
}

inline double PdfPow(double a, double b, double t, int yobs)
{
	double p;
	double y;
	p=ProbPow(a,b,t);
	y=pow(p,yobs)*pow((1-p),(SN-yobs));
	return y;
}

inline double PdfHyperbolic(double a, double b, double t, int yobs)
```

```
{
	double p;
	double y;
	p=ProbHyperbolic(a,b,t);
	y=pow(p,yobs)*pow((1-p),(SN-yobs));
	return y;
}

inline double KLPQ(double (*funcpprob)(double, double, double, double, double,
double), double (*funcqprob)(double, double, double), double powa,double powb,
double expa, double expb, double w, double aq, double bq, double t)
{
	double p,q;
	double ans;
	p=funcpprob(powa,powb,expa,expb,w,t);
	q=funcqprob(aq,bq,t);
	ans=SN*p*log(p/q)+SN*(1-p)*log((1-p)/(1-q));
	return ans;
}



inline double overtKLPQ(double (*funcpprob)(double, double, double, double,
double, double), double (*funcqprob)(double, double, double), double powa,double
powb, double expa, double expb, double w, double aq, double bq)
{
	double t,u;
	t=0.001;
	u=0;
	while(t<100)
	{
		u=u+KLPQ(funcpprob,funcqprob,powa,powb,expa,expb,w,aq,bq,t);
		t=t+0.1;//1000
	}
	u=u/1000;
	return u;
}

inline double localKLPQ(double (*funcpprob)(double, double, double, double,
double, double), double (*funcqprob)(double, double, double), double powa,double
powb, double expa, double expb, double w, double aq, double bq, MATRIXTYPE design)
{
	double u;
	int j=0;

	u=0;
	while(j<10)
	{

	u=u+KLPQ(funcpprob,funcqprob,powa,powb,expa,expb,w,aq,bq,design[j][0]);
		j++;
	}
	u=u/10;
	return u;
}
```

RNGs.h

```
//This part includes functions for generating random numbers from some common
//probability distributions.

const double PI=3.14159265358979323846;
inline double RandomDouble() //GENERATES RANDOM NUMBER BETWEEN 0 AND 1
{
	//return (double(rand())/double(32767));
	return TRandom();
}

inline MATRIXTYPE rand(int np)    //COMPUTES A VECTOR OF STANDARD NORMAL NUMBERS
{
	MATRIXTYPE Result(1,np);
	Result.Random(&RandomDouble);
	return Result;
};

inline double RandNorm()  //COMPUTES A NUMBER FROM THE STANDARD NORMAL DIST.
{
	double U1, U2, x;
	U1=RandomDouble();
	U2=RandomDouble();
	x=sqrt(-2*log(U1)) * cos(2*PI*U2);
	return x;
};

inline MATRIXTYPE randn(int np)   //COMPUTES A VECTOR OF STANDARD NORMAL NUMBERS
{
	MATRIXTYPE Result(1,np);
	Result.Random(&RandNorm);
	return Result;
};
inline MATRIXTYPE RandBeta(int a, int b, int n) //COMPUTES TWO VECTORS OF BETA
NUMBERS
{
	MATRIXTYPE beta(n,1);

	for(int ii=0; ii<n; ii++)
	{
		double gama=1, gamb=1;
		int i=0, j=0;
		while (i<a)
		{
			gama=gama*RandomDouble();
			i++;
		};
		while (j<b)
		{
			gamb=gamb*RandomDouble();
			j++;
		};
```

```
            gama=log(gama);
            gamb=log(gamb);
            beta[0][ii]=gama/(gama+gamb);
      };
      return beta;
};
inline double RandGamma(int a, double b) //COMPUTES A BETA(A,B) NUMBER
{
      double gamma;

      double gam=1;
      int i=0;
      while (i<a)
      {
            gam=gam*RandomDouble();
            i++;
      };
   gamma=(-1/b)*log(gam);
      return gamma;
};
inline MATRIXTYPE randg(int a, double b, int n)
{
      MATRIXTYPE gamma(n,1);

      for(int ii=0; ii<n; ii++)
      {
            gamma[ii][0]=RandGamma(a,b);
      };
      return gamma;
};

inline MATRIX<int> binornd(int NUM, MATRIXTYPE &P)
{
    MATRIX<int> Result(P.GetHeight(), P.GetWidth());
    int l;
    int h=P.GetHeight();
    int w=P.GetWidth();
    for (int i=0; i<h; i++)
    {
        for (int j=0; j<w; j++)
        {
            l=0;
            Result[i][j]=0;
            while (l<NUM)
            {
                if (RandomDouble()<P[i][j])
                {
                    Result[i][j]++;
                }
                l++;
            };
        }
    }
    return Result;
};
```

```
inline MATRIX<int> multinorm(int NUM, MATRIXTYPE wgt)
{
	int i=0;
	MATRIX<int> Result(1, wgt.GetWidth());
	MATRIX<int> z;
	MATRIXTYPE onesNC(1,wgt.GetWidth());
	MATRIXTYPE onesN(NUM,1);
	MATRIXTYPE randN(NUM,1);
	MATRIXTYPE cprb;
	onesNC=1;
	onesN=1;

	cprb=wgt.CumSum();
	randN.Random(&RandomDouble,1,0);
	z=((randN*onesNC) > (onesN*cprb)).SumByRow();
	//z=z.SumByRow();
	Result=0;

	while (i<NUM)
	{
		Result[0][z[0][i]]++;
		i++;
	};
	return Result;
};
```