Queen's Economics Department Working Paper No. 1201

A Practitioner's Guide to Bayesian Estimation of Discrete Choice Dynamic Programming Models

Susumu Imai
Queen's University

Andrew Ching
University of Toronto

Masakazu Ishihara
University of Toronto

Neelan Jain
Northern Illinois University

Department of Economics
Queen's University
94 University Avenue
Kingston, Ontario, Canada
K7L 3N6

4-2009

A Practitioner's Guide to Bayesian Estimation of Discrete Choice Dynamic Programming Models*

Andrew Ching
Rotman School of Management
University of Toronto

Susumu Imai
Department of Economics
Queen's University

Masakazu Ishihara
Rotman School of Management
University of Toronto

Neelam Jain
Department of Economics
Northern Illinois University

This draft: April 5, 2009

*This is work-in-progress. Comments are welcome.

Abstract

This paper provides a step-by-step guide to estimating discrete choice dynamic programming (DDP) models using the Bayesian Dynamic Programming algorithm developed in Imai, Jain and Ching (2008) (IJC). The IJC method combines the DDP solution algorithm with the Bayesian Markov Chain Monte Carlo algorithm into a single algorithm, which solves the DDP model and estimates its structural parameters simultaneously. The main computational advantage of this estimation algorithm is the efficient use of information obtained from the past iterations. In the conventional Nested Fixed Point algorithm, most of the information obtained in the past iterations remains unused in the current iteration. In contrast, the Bayesian Dynamic Programming algorithm extensively uses the computational results obtained from the past iterations to help solving the DDP model at the current iterated parameter values. Consequently, it significantly alleviates the computational burden of estimating a DDP model. We carefully discuss how to implement the algorithm in practice, and use a simple dynamic store choice model to illustrate how to apply this algorithm to obtain parameter estimates.

1 Introduction

In economics and marketing, there is a growing empirical literature which studies choice of agents in both the demand and supply side, taking into account their forward-looking behavior. A common framework to capture consumers or firms forward-looking behavior is discrete choice dynamic programming (DDP) framework. This framework has been applied to study manager's decisions to replace old equipments (e.g., Rust 1987), career decision choice (e.g., Keane and Wolpin 1997; Diermier, Merlo and Keane 2005), choice to commit crimes (Imai and Krishna 2004), dynamic brand choice (e.g., Erdem and Keane 1996; Gönül and Srinivasan 1996), dynamic quantity choice with stockpiling behavior (e.g., Erdem, Imai and Keane 2003; Sun 2005; Hendel and Nevo 2006), new product/technology adoption decisions (Ackerberg 2003; Song and Chintagunta 2003; Crawford and Shum 2005; Yang and Ching 2008), new product introduction decisions (Hitsch 2006), etc. Although the framework provides a theoretically tractable way to model forward-looking incentives, and this literature has been growing, it remains small relative to the literature that models choice using a static reduced form framework. This is mainly due to two obstacles of estimating this class of models: (i) the curse of dimensionality problem in the state space, putting a constraint on developing models that match the real world applications; (ii) the complexity of the likelihood/GMM objective function, making it difficult to search for the global maximum/minimum when using classical approach to estimate them. Several studies have proposed different ways to approximate the dynamic programming solutions, and reduce the hurdle due to the curse of dimensionality problem (e.g., Keane and Wolpin 1994; Rust 1997; Hotz and Miller 1993; Aguirreagabiria and Mira 2002; Ackerberg 2001).[1] Nevertheless, little progress has been made in handling the complexity of the likelihood function from the DDP models. A typical approach is to use different initial values to re-estimate the model, and check which set of parameter estimates gives the highest likelihood value. However, without knowing the exact shape of the likelihood function, it is often difficult to confirm whether the estimated parameter vector indeed gives us the global maximum.

In the past two decades, Bayesian Markov Chain Monte Carlo (MCMC) approach has provided a tractable way to simulate the posterior distribution of parameter vectors for complicated static discrete choice models, making the posterior mean an attractive estimator compared with classical point estimates in that setting (Albert and Chib 1993; McCulloch and Rossi 1994; Allenby and Lenk 1994; Allenby 1994; Rossi et al. 1996; Allenby and Rossi 1999). However, researchers seldom use the Bayesian approach to estimate DDP models. The main problem is that the Bayesian MCMC approach typically requires many more iterations than classical approach to achieve convergence. In each simulated draw of a parameter vector, the DDP model needs to be solved to calculate the likelihood function. As a result, the computational burden of solving a DDP model has essentially ruled out the Bayesian approach except for very simple models, where the solution of the model can be solved very quickly or there exists a close form solution (e.g., Lancaster 1997).

Recently, Imai, Jain and Ching (2008) (IJC) propose a new modified MCMC algorithm

[1]Geweke and Keane (2002) proposed to use a flexible polynomial to approximate the future component of the Bellman equation. Their approach allowed them to conduct Bayesian inference on the structural parameters of the current payoff functions and the reduced form parameters of the polynomial approximations. However, since it completely avoids solving for the DDP model and fully specifying it, their estimation results are not efficient and policy experiments cannot be conducted under their approach.

to reduce the computational burden of estimating infinite horizon DDP models using the Bayesian approach. This method combines the DDP solution algorithm with the Bayesian MCMC algorithm into a single algorithm, which solves the DDP model and estimates its structural parameters simultaneously. In the conventional Nested Fixed Point algorithm, most of the information obtained in the past iterations remains unused in the current iteration. In contrast, the IJC algorithm extensively uses the computational results obtained from the past iterations to help solving the DDP model at the current iterated parameter values. This new method is potentially superior to prior methods because (1) it significantly reduces the computational burden of solving for the DDP model in each iteration, and (2) it produces the posterior distribution of parameter vectors, and the corresponding solutions for the DDP model – this avoids the need to search for the global maximum of a complicated likelihood function.

This paper provides a step-by-step guide to estimating discrete choice dynamic programming (DDP) models using the IJC method. We carefully discuss how to implement the algorithm in practice, and use a simple dynamic store choice model to illustrate how to apply this algorithm to obtain parameter estimates. Our goal is to reduce the costs of adopting this new method and expand the toolbox for researchers who are interested in estimating DDP models. The rest of the paper is organized as follows. In section 2, we present a dynamic store choice model, where each store offers its own reward programs. In section 3, we present the IJC method and explain how to implement it to obtain parameter estimates of this model. We also discuss the practical aspects of using the IJC method. In section 4 we use the estimation results of the dynamic store choice model to demonstrate certain properties of the IJC estimation method. Section 5 discusses how to extend IJC algorithm to (i) conduct policy experiments, and (ii) allow for continuous state variables. We also comment on the choice of kernels. Section 6 is the conclusion.

2 The Model

2.1 The Basic Framework

Suppose that there are two supermarkets in a city ($j = 1, 2$). Each store offers a stamp card, which can be exchanged for a gift upon completion. Consumers get one stamp for each visit with a purchase.

Reward programs at the two supermarkets differ in terms of (i) the number of stamps required for a gift ($\bar{S}_j$), and (ii) the mean value of the gift (G_j). Consumers get a gift in the same period (t) that they complete the stamp card. Once consumers receive a gift, they will start with a blank stamp card again in the next period.

Let p_{ijt} be the price that consumer i pays in supermarket j at time t. We assume that prices of store j in each period are drawn from an iid normal distribution, $N(\bar{p}, \sigma_p^2)$, which is common across stores. Also, we assume that this price distribution is known to consumers. Let $s_{ijt} \in \{0, 1, \ldots, \bar{S}_j - 1\}$ denote the number of stamps collected for store j in period t before consumers make a decision. Note that s_{ijt} does not take the value $\bar{S}_j$ because of our assumption that consumers get a gift in the same period that they complete the stamp card.

Consumer i's single period utility of visiting supermarket j in period t at $s_{it} = (s_{i1t}, s_{i2t})$ is given by

$$U_{ijt}(s_{it}) = \begin{cases} \alpha_j + \gamma p_{ijt} + G_{ij} + \epsilon_{ijt} & \text{if } s_{ijt} = \bar{S}_j - 1 \\ \alpha_j + \gamma p_{ijt} + \epsilon_{ijt} & \text{otherwise,} \end{cases}$$

where α_j is the consumer loyalty (or brand equity) for store j, γ is the price sensitivity,

G_{ij} is consumer i's valuation of gift for store j, and ϵ_{ijt} is the idiosyncratic error term. We assume ϵ_{ijt} is extreme-value distributed. G_{ij} is assumed to be normally distributed around G_j with the standard deviation σ_{G_j}. In each period, consumers may choose not to go shopping. The single period mean utility of no shopping is normalized to zero, i.e., $U_{i0t} = \epsilon_{i0t}$.

Consumer i's objective is to choose a sequence of store choices to maximize the sum of the present discounted future utility:

$$\max_{\{d_{ijt}\}_{t=1}^{\infty}} E\left[\sum_{t=1}^{\infty} \beta^{t-1} d_{ijt} U_{ijt}(s_{it})\right]$$

where $d_{ijt} = 1$ if consumer i chooses store j in period t and $d_{ijt} = 0$ otherwise. β is the discount factor. The evolution of state, s_{it}, is deterministic and depends on consumers' choice. Given the state s_{ijt}, the next period state, s_{ijt+1}, is determined as follows:

$$s_{ijt+1} = \begin{cases} s_{ijt} + 1 & \text{if } s_{ijt} < \bar{S}_j - 1 \text{ and purchase at store } j \text{ in period } t \\ 0 & \text{if } s_{ijt} = \bar{S}_j - 1 \text{ and purchase at store } j \text{ in period } t \\ s_{ijt} & \text{if purchase at store } -j \text{ or no shopping in period } t \end{cases} \tag{1}$$

Let θ be the vector of parameters. Also, define $s_i = (s_{i1}, s_{i2})$, $p_i = (p_{i1}, p_{i2})$, and $G_i = (G_{i1}, G_{i2})$. In state s_i, the Bellman's equation for consumer i is given by

$$\begin{aligned} V(s_i, p_i; G_i, \theta) &\equiv E_\epsilon \max\{V_0(s_i; \theta) + \epsilon_{i0}, V_1(s_i, p_{i1}; G_i, \theta) + \epsilon_{i1}, V_2(s_i, p_{i2}; G_i, \theta) + \epsilon_{i2}\} \\ &= \log(\exp(V_0(s_i; \theta)) + \exp(V_1(s_i, p_{i1}; G_i, \theta)) + \exp(V_2(s_i, p_{i2}; G_i, \theta))), \end{aligned} \tag{2}$$

where the second equality follows from the extreme value assumption on ϵ. The alternative-specific value functions are written as: For $j = 1, 2$,

$$V_j(s_i, p_{ij}; G_i, \theta) = \begin{cases} \alpha_j + \gamma p_{ij} + G_{ij} + \beta E_{p'}[V(s', p'; G_i, \theta)] & \text{if } s_{ij} = \bar{S}_j - 1, \\ \alpha_j + \gamma p_{ij} + \beta E_{p'}[V(s', p'; G_i, \theta)] & \text{otherwise.} \end{cases} \tag{3}$$

$$V_0(s_i; \theta) = \beta E_{p'}[V(s', p'; G_i, \theta)]$$

where the state transition from s_i to s' follows (1), and the expectation with respect to p' is defined as

$$E_{p'}[V(s', p'; G_i, \theta] = \int V(s', p'; G_i, \theta) dF(p').$$

The parameters of the model are α_j (store loyalty), G_j (consumers' mean value of the gift offered by store j), σ_{G_j} (standard deviation of G_{ij}), γ (price sensitivity), β (discount factor), $\bar{p}$ (mean price common across stores), σ_p (standard deviation of prices).

Hartmann and Viard (2008) estimated a dynamic model with reward programs that is similar to the one here. The main differences are (1) we allow for two stores with different reward programs in terms of $(G_j, \bar{S}_j)$ while they considered one store (golf club); (2) we estimate the discount factor (i.e., β) while they fixed it according to the interest rate. The general dynamics of this model is also more complicated than the one used in IJC for Monte Carlo exercises. The model here has two endogenous state variables (s_1, s_2), while the dynamic firm entry-exit decision model used in IJC has one exogenous state variable (capital stock). However, IJC consider a normal error term, which is more general than the extreme value error term we assume here. We consider the extreme value error term because (1) it is quite common that researchers adopt this distributional assumption when estimating a DDP model, (2) our analysis here would complement that of IJC.

2.2 Identification

The main dynamics of the model is the intertemporal trade-off created by the reward program. Suppose that a consumer is closer to the completion of the stamp card for store 1, but the price is lower in store 2 today. If the consumer chooses store 2 based on the lower price, he or she will delay the completion of the stamp card for store 1. If the consumer take the future into account, the delay will lower the present discounted value of the reward. Thus he/she will have an incentive to keep shopping at store 1 even though prices at store 2 is lower. Moreover, such an incentive should depend on the discount factor.

This dynamic trade-off suggests that the empirical choice frequency of visiting the stores across states should allow us to pin down the discount factor. To illustrate this point, we consider a model of homogeneous consumers with only one store and an outside option and simulate choice probabilities for different discount factors. In this exercise, we set $\alpha_1 = -2$, $\gamma = 0$, $G_1 = 3$, and $\bar{S}_1 = 5$. Figure 1 shows how the choice probability of visiting the store changes across states (no. of stamps collected) for different discount factors ($\beta = 0, 0.5, 0.75, 0.9, 0.999$). When $\beta = 0$, the choice probability is purely determined by the single period utility. Thus, the choice probability is flat from $s = 0$ to $s = 3$. At $s = 4$, consumers receive the gift thus the choice probability jumps up. Another extreme case is when β is close to one ($\beta = 0.999$). Interestingly, the choice probabilities are essentially constant across states, and higher than those of $\beta = 0$ for $s = 0, ..., 3$. But at $s = 4$, the choice probability for $\beta = 0.999$ is smaller than that for $\beta = 0$.

Figure 2 shows how the choice probabilities at each state change with β. In general, as we increase β, we observe three predictions: (i) when $s = 4$, the store choice probability monotonically decreases; (ii) when $s < 3$, the store choice probability monotonically increase; (iii) when $s = 3$, the store choice probability first increases and then decreases. Why do we have prediction (i)? First, we note that $E_pV(s)$ always increases with β for all s, but the extent of the increase would differ across s. In particular, $EV(s = 4)$ is much more responsive than $EV(s = 0)$ to an increase in β. This is because a consumer is much closer to getting a reward at $s = 4$. This shows that when β increases, the value of choosing the outside option (depending on $EV(s = 4)$) will increase faster than that of choosing to visit the store (depending on $EV(s = 0)$). As a result, the store choice probability decreases with β. Prediction (ii) is mainly due to the fact that the incentive to earn a reward increases with β. To understand prediction (iii), we note that in general an increase in β has two effects on store choice. On the one hand, it increases the incentive to earn a reward as the consumer puts more weight in the future. On the other hand, a larger β also implies that the consumer is more patient, and consequently, becomes less concern about waiting (i.e., the costs of waiting become smaller). The interaction between these two forces explains prediction (iii). When β first increases from zero, the former force (i.e., the incentive to earn a reward earlier) dominates. But as β becomes closer to 1, the later force (i.e., the patience factor) catches up. Lastly, it should be noted that the choice probabilities become flat when β is very close to one. This is because the difference between these two forces essentially becomes constant across s, when β approaches one.

The above discussion also suggests that the identification of α_j and G_j crucially depends on the value of β. When β is close to 1, it is very hard to identify them separately because they play very similar role in determining the choice probabilities. But when β moves away from 1, it becomes much easier to identify them.[2]

[2]Hartmann and Viard (2008) also discussed how the discount factor would affect the pattern of choice probabilities. However, because they take the intrinsic discount factor as exogenously given (determined by the interest rate), they argue that such an effect would happen through the "artificial" discount factor, which depends on how frequent a customer visits a store (determined by α_j here).

2.3 Numerical Solution of the Model

We will consider how to solve the model without consumer heterogeneity in G_{ij} first. Solving the model with consumer heterogeneity is a straightforward extension. To solve the model without consumer heterogeneity (i.e., $G_{ij} = G_j$ for all i and $j = 1, 2$) numerically, we will

1. Make M draws of $\{p_j^m\}_{m=1}^M$ from the price distribution function, $N(\bar{p}, \sigma_p^2)$, for $j = 1, 2$. We fix these draws below.

2. Start with an initial guess of the Emax functions, e.g., setting $V^0(s, p; \theta) = 0, \forall s, p$. Suppose that we know V^l, where l is the number of past iterations. We will discuss how to obtain V^{l+1}.

3. Substitute $\{p^m\}_{m=1}^M$ into $V^l(s, p; \theta)$, and then take the average across p^m's to obtain a Monte Carlo approximation of $E_{p'} V^l(s', p'; \theta), \forall s'$.

4. Substitute these approximated expected future value functions into the Bellman operator and obtain $V^{l+1}(s, p; \theta), \forall s, p$.

5. Repeat step 3-4 until $E_p V^{l+1}(., p; \theta)$ converges.

For the model with consumer heterogeneity in G_{ij}, we will need to compute $V^{l+1}(s, p; G_i, \theta)$ for each i.

3 Estimation Method

3.1 IJC algorithm

It is well-known that when using maximum likelihood or Bayesian MCMC to estimate discrete choice dynamic programming models, the main computational burden is that the value functions need to be solved in each set of trial parameter vector (for maximum likelihood), or each set of random draw of parameter vector (for Bayesian MCMC). Since both procedures, in particular Bayesian MCMC, require many iterations to achieve convergence, a typical nested fixed point algorithm will need to repeatedly apply the re-iteration procedure outlined above to solve for the value functions.[3] As a result, the computational burden is so large that even a very simple discrete choice dynamic programming model cannot be estimated using standard Bayesian MCMC methods.

The IJC algorithm relies on two insights to reduce the computational burden of each MCMC iteration: (1) It could be quite wasteful to compute the value function exactly before the markov chain converges to the true posterior distribution. Therefore, the IJC algorithm proposes to "partially" solve for the Bellman equation for each parameter draw (at the minimum, only apply the Bellman operator once in each iteration). (2) The value functions evaluated at the past MCMC draws of parameters contain useful information about the value functions at the current draw of parameters, in particular, for those evaluated within a neighborhood of the current parameter values. However, the traditional nested fixed point algorithm hardly makes use of them. Therefore, IJC propose to replace the contraction mapping procedure of solving the value functions with a weighted average

[3]It should be noted that Bayesian MCMC algorithm generally needs to be run 10,000 to 30,000 iterations to obtain enough draws for the posterior distributions.

of the pseudo-value functions obtained as past outcomes of the estimation algorithm. In IJC, the weight depends on the distance between the past parameter vector draw and the current one – the shorter the distance, the higher the weight. The basic intuition is that the value function is continuous in the parameter space. Therefore, it is possible to use the past value functions to form a non-parametric estimate of the value function evaluated at the current draw of parameter values. Such a non-parametric estimate is usually computationally much cheaper than the standard contraction mapping procedure, in particular for β close to 1. Combining these two insights, IJC dramatically reduce the computational burden of each iteration in the Bayesian MCMC algorithm. This modified procedure differs from the standard nested fixed point algorithm in an important aspect: instead of solving the model and search for parameters alternately, it solves and estimates the model simultaneously.

In the context of the reward program example without consumer heterogeneity, the outputs of the algorithm in each iteration r include $\{\theta^r, E_p\tilde{V}^r(s,p;\theta^r)\}$, where $\tilde{V}^r$ is the pseudo-value function. To obtain these outputs, IJC make use of the past outcomes of the estimation algorithm, $H^r = \{\theta^l, E_p\tilde{V}^l(s,p;\theta^l)\}_{l=r-N}^{r-1}$. We will now extend Bellman equations (2)-(3) to illustrate the pseudo-value functions defined in IJC.

The pseudo-value functions are defined as follows. To simplify notations, we drop the i subscript for s and p. For each s,

$$\tilde{V}^r(s,p;\theta^r) = \log(\exp(\tilde{V}_0^r(s;\theta^r)) + \exp(\tilde{V}_1^r(s,p_1;\theta^r)) + \exp(\tilde{V}_2^r(s,p_2;\theta^r))), \qquad (4)$$

where for $j = 1, 2$,

$$\tilde{V}_j^r(s, p_j; \theta^r) = \begin{cases} \alpha_j - \gamma p_j + G_j + \beta \hat{E}_{p'}^r V(s', p'; \theta^r) & \text{if } s_j = N_j - 1, \\ \alpha_j - \gamma p_j + \beta \hat{E}_{p'}^r V(s', p'; \theta^r) & \text{otherwise,} \end{cases} \tag{5}$$
$$\tilde{V}_0^r(s; \theta^r) = \beta \hat{E}_{p'}^r V(s', p'; \theta^r).$$

The approximated Emax functions are defined as the weighted average of the past pseudo-value functions obtained from the estimation algorithm. For instance, $\hat{E}_p^r V(s, p; \theta^r)$ can be constructed as follows:

$$\hat{E}_p^r V(s, p; \theta^r) = \sum_{l=r-N}^{r-1} E_p \tilde{V}^l(s, p; \theta^l) \frac{K_h(\theta^r - \theta^l)}{\sum_{k=r-N}^{r-1} K_h(\theta^r - \theta^k)}, \tag{6}$$

where $K_h(.)$ is a Gaussian kernel with bandwidth h. To obtain $E_p \tilde{V}^r(s, p; \theta^r)$, we simulate a set of prices, $\{p^m\}_{m=1}^M$, from the price distribution, $N(\underline{p}, \sigma_p^2)$, and evaluate $\tilde{V}^r(s, p^m; \theta^r)$ using (4) and (5) and,

$$E_p \tilde{V}^r(s, p; \theta^r) = \sum_{m=1}^M \tilde{V}^r(s, p^m; \theta^r), \tag{7}$$

IJC show that by treating the pseudo-value function as the true value function in a MCMC algorithm, and applying it to estimate a dynamic discrete choice problem with discrete state space, the modified MCMC draws of θ^r converge to the true posterior distribution uniformly.

It should be highlighted that the approximated Emax function defined in (6) is the key innovation of IJC. In principles, this step is also applicable in classical estimation methods such as GMM and Maximum Likelihood.[4] However, there are at least several advantages of implementing IJC's pseudo-value function approach in Bayesian estimation. First, the non-parametric approximation in (6) would be more efficient if the past pseudo-value

[4]Brown and Flinn (2006) extend the implementation of this key step in estimating a dynamic model of marital status choice and investment in children using the method of simulated moments.

functions are evaluated at θ^l's that are randomly distributed around θ^r. This can be naturally achieved by the Bayesian MCMC algorithm. On the contrary, classical estimation methods typically require minimizing/maximizing an objective function. Commonly used minimization/maximization routines (e.g., BHHH, quasi-Newton methods, etc.) tend to search over parameter spaces along a particular path. Consequently, we believe that the approximation step proposed by IJC should perform better under the Bayesian MCMC approach.[5] Second, in the presence of unobserved consumer heterogeneity, it is common that the likelihood function is multi-modal even for static choice problems. In this situation, Bayesian posterior means often seem to be better estimators of true parameter values than are classical point estimates. This is because in practice, accurately simulating a posterior is in many cases much easier than finding the global maximum/minimum of a complex likelihood/GMM objective function.

Now we turn to discuss how to implement the IJC method to estimate the dynamic store choice model present here. We consider two versions of the model: (i) without unobserved consumer heterogeneity, (ii) with unobserved consumer heterogeneity.

3.2 Implementation of the IJC algorithm

In this subsection, we discuss how to estimate the dynamic store choice model using the IJC algorithm. The steps are similar to the standard Metropolis-Hastings algorithm, except that we use (6) to calculate the Emax. Let $I_{buy,ijt}$ be an indicator function for

[5]A stochastic optimization algorithm, simulated annealing, has recently gained some attention to handle complicated objective function. This algorithm is an adaptation of the Metropolis-Hastings algorithm (Kirkpatrick et al. 1983; Černý 1985). The IJC method should also be well-suited when incorporating with simulated annealing for the classical estimation approach. However, we should note that before a researcher starts the estimation, this method requires him/her to choose a "cooling" rate. The ideal cooling rate cannot be determined a priori. In the MCMC-based Bayesian algorithm, one does not need to deal with this nuisance parameter.

purchasing at store j by consumer i at time t, and $p_{it} = (p_{i1t}, p_{i2t})$ be the price vector for consumer i at store j at time t. We use $(I^d_{buy,ijt}, p^d_{ijt})$ to denote the observed data.

Our focus is to provide a step-by-step implementation guide and discuss the practical aspects of the IJC algorithm. Once the readers understand implementation of the IJC algorithm in this simple example, they should be able to extend it to other more complicated settings.

3.2.1 Homogeneous Consumers

We first present the implementation of the IJC algorithm when consumers are homogeneous in their valuations of G_j (i.e., $\sigma_{G_j} = 0$ for $j = 1, 2$). The vector of parameters to be estimated is $\theta = (\alpha_1, \alpha_2, G_1, G_2, \gamma, \beta)$.

The IJC algorithm generates a sequence of MCMC draw of $\theta^r, r = 1, 2, ...$ The algorithm modifies the Metropolis-Hastings, and involves drawing a candidate parameter θ^{*r} in each iteration. For each iteration (r),

1. Start with a history of past outcomes from the algorithm,

$$H^r = \{\{\theta^{*l}, E_p\tilde{V}^l(., p; \theta^{*l})\}_{l=r-N}^{r-1}, \rho^r(\theta^{r-1})\}$$

   where N is the number of past iterations used for Emax approximation; $E_p\tilde{V}^l$ is the expected pseudo-value functions w.r.t. p; $\rho^r(\theta^{r-1})$ is the pseudo-likelihood of the data conditional on $\hat{E}^r_pV(., p; \theta^{r-1})$.

2. Draw θ^{*r} (candidate parameter vector) from a proposal distribution $q(\theta^{r-1}, \theta^{*r})$.

3. Compute the pseudo-likelihood conditional on θ^{*r} based on the approximated Emax functions. Then we determine whether or not to accept θ^{*r} based on the acceptance

probability, $\min(\frac{\pi(\theta^{*r})\cdot\rho^r(\theta^{*r})\cdot q(\theta^{*r},\theta^{r-1})}{\pi(\theta^{r-1})\cdot\rho^r(\theta^{r-1})\cdot q(\theta^{r-1},\theta^{*r})},1)$. Essentially, we apply the Metropolis-Hastings algorithm here by treating the pseudo-likelihood as the true likelihood. If accept, set $\theta^r = \theta^{*r}$; otherwise, set $\theta^r = \theta^{r-1}$. To obtain $\rho^r(\theta^{*r})$, we need to calculate $\hat{E}_p^r V(.,p;\theta^{*r})$, which is obtained using the weighted average of the past expected pseudo-value functions: $\{E_p\tilde{V}^l(.,p;\theta^{*l})\}_{l=r-N}^{r-1}$. The weight of each past expected pseudo-value function is determined by Gaussian kernels.

$$\hat{E}_p^r V(s,p;\theta^{*r}) = \sum_{l=r-N}^{r-1} E_p\tilde{V}^l(s,p;\theta^{*l})\frac{K_h(\theta^{*r}-\theta^{*l})}{\sum_{k=r-N}^{r-1}K_h(\theta^{*r}-\theta^{*k})}.$$

4. Computation of pseudo-value function, $E_p\tilde{V}^r(s,p;\theta^{*r})$.

   (a) Make M draws of prices, $\{p^m\}_{m=1}^M$, from the price distribution.

   (b) Compute $\tilde{V}_0^r(s;\theta^{*r})$, $\tilde{V}_1^r(s,p_1^m;\theta^{*r})$ and $\tilde{V}_2^r(s,p_2^m;\theta^{*r})$, using the approximated Emax functions computed in Step 3.

   (c) Given $\tilde{V}_0^r(s;\theta^{*r})$, $\tilde{V}_1^r(s,p_1^m;\theta^{*r})$ and $\tilde{V}_2^r(s,p_2^m;\theta^{*r})$, obtain the pseudo-value function, $\tilde{V}^r(s,p^m;\theta^{*r})$ (see (4)). By averaging $\tilde{V}^r(s,p^m;\theta^{*r})$ across p^m's, we integrate out prices and obtain $E_p\tilde{V}^r(s,p;\theta^{*r})$.

5. Compute the pseudo-likelihood, $\rho^{r+1}(\theta^r)$, based on a updated set of past pseudo-value functions.[6]

6. Go to iteration $r+1$.

Note that when implementing the IJC algorithm, we propose to store $\{\theta^{*l}, E_p\tilde{V}^l(s,p;\theta^{*l})\}_{l=r-N}^{r-1}$ instead of $\{\theta^l, E_p\tilde{V}^l(s,p;\theta^l)\}_{l=r-N}^{r-1}$. If we store $\{\theta^l, E_p\tilde{V}^l(s,p;\theta^l)\}_{l=r-N}^{r-1}$, there may be a

[6]Two points should be noted here. First, in each iteration, there is a new pseudo-value function evaluated at θ^{*r}. Second, this step needs not be carried out in a full-solution based Bayesian MCMC algorithm.

significant portion of θ^l's that are repeated because the acceptance rate of the M-H step is usually set at around $\frac{1}{3}$. In order to conduct the non-parametric approximation for the expected future value, it is useful to have a set of $E_p\tilde{V}^l$'s that span the parameter spaces. Since θ^{*l}'s are drawn from a candidate generating function, it is much easier for us to achieve this goal by storing $E_p\tilde{V}^l$'s at θ^{*l}'s. Moreover, for each candidate parameter draw, θ^{*r}, we need to evaluate the expected future payoffs, $\hat{E}_p^r V(., p; \theta^{*r})$, to form the likelihood. As we have shown above, it will only take an extra step to obtain $E_p\tilde{V}^r(., p; \theta^{*r})$. So storing $\{E_p\tilde{V}^r(., p; \theta^{*l})\}_{l=r-N}^{r-1}$ will impose little extra costs.

The above description of the implementation of the algorithm is slightly different from IJC, who propose to draw only one unobserved error term in each iteration. In the current application, that would be equivalent to drawing one price shock in each iteration, p^l, and store $H^r = \{\{\theta^{*l}, \tilde{V}^l(., p^l; \theta^{*l})\}_{l=r-N}^{r-1}, \rho^r(\theta^{r-1})\}$. The main difference between these two approaches is that the original IJC proposes to store $\tilde{V}^l$ instead of $E_p\tilde{V}^l$, and the Emax approximation is,

$$\hat{E}_p^r V(s, p; \theta^{*r}) = \sum_{l=r-N}^{r-1} \tilde{V}^l(s, p^l; \theta^{*l}) \frac{K_h(\theta^{*r} - \theta^{*l})}{\sum_{k=r-N}^{r-1} K_h(\theta^{*r} - \theta^{*k})}.$$

The advantage of the original approach is that it saves some time in computing $\hat{E}_p\tilde{V}^r$. However, we believe that the approach proposed here would allow us to to achieve the same level of precision in terms of integrating out prices by using a smaller N. As a result, it would also save us some time in computing the weighted average.

We should also note that in the present example where we assume prices are observed, one can use the observed prices as random realizations in computing $E_p\tilde{V}^r(s, p; \theta^r)$, provided that there are sufficient number of observations for each s. The advantage of using the observed prices is that the pseudo-value functions are by-products of the likelihood function computation. However, this is only possible when we have a reasonably large number of observations for each state.

We also note that in step 5, it may not be worthwhile to compute the pseudo-likelihood, $\rho^{r+1}(\theta^r)$, in each iteration. If we accept θ^{*r}, $\rho^r(\theta^{*r})$, which is calculated in the M-H step (i.e., step 3), can be used as a proxy for $\rho^{r+1}(\theta^r)$, which is computed in step 5. Note that their calculations only differ in terms of one past pseudo-value function. If we reject θ^{*r}, we set $\theta^r = \theta^{r-1}$. So we could use $\rho^r(\theta^{r-1})$ as a proxy for $\rho^{r+1}(\theta^r)$, and only compute $\rho^{r+1}(\theta^r)$ once every several successive rejections. Our experiences suggest that we can obtain a fairly decent gain in computational time if we use this approach.

3.2.2 Heterogeneous Consumers

We now present the implementation of the IJC algorithm when consumers have heterogeneous valuations for the reward ($\sigma_{G_j} > 0$). The vector of parameters to be estimated is $\theta = (\theta_1, \theta_2)$, where $\theta_1 = (\alpha_1, \alpha_2, \gamma, \beta)$ and $\theta_2 = (G_1, G_2, \sigma_{G_1}, \sigma_{G_2})$. We incorporate the Bayesian approach for random-coefficient models into the estimation steps for homogeneous case. We use a normal prior on G_j and an inverted gamma prior on σ_{G_j}.

We use the Metropolis-Hastings algorithm to draw G^l_{ij}, which is consumer i's valuation of reward at store j from the population distribution of G_{ij}. This draw is regarded as an individual specific parameter. As a result, conditional on G_i, the value functions do not depend on θ_2.

Each MCMC iteration mainly consists of three blocks.

(i) Draw $G^r_j \sim f_G(G_j|\sigma^{r-1}_{G_j}, \{G^{r-1}_{ij}\}^I_{i=1})$ and $\sigma^r_{G_j} \sim f_{\sigma_G}(\sigma^r_{G_j}|G^r_j, \{G^{r-1}_{ij}\}^I_{i=1})$ for $j = 1, 2$

(the parameters that capture the distribution of G_{ij} for the population) where f_G and f_{σ_G} are the posterior distributions.

(ii) Draw individual parameters G^r_{ij} by the independent Metropolis-Hastings algorithm with $N(G^r_j, (\sigma^r_{G_j})^2)$ as a proposal distribution for all i and $j = 1, 2$.

(iii) Draw $\theta^r_1 \sim f_{\theta_1}(.|G^r_i)$ using the random-walk Metropolis-Hastings algorithm.

The estimation steps are as follows. Step 2-3 belong to block (i), step 4 belongs to block (ii) and step 5 belongs to block (iii). For each MCMC iteration (r),

1. Start with

$$H^r = \{\{\theta^{*l}, G^l_{i^*}, E_p\tilde{V}^l(., p; G^l_{i^*}, \theta^{*l}_1)\}^{r-1}_{l=r-N}, \{\rho^r_i(G^{r-1}_i, \theta^{r-1}_1)\}^I_{i=1}\}$$

   where I is the number of consumers; N is the number of past iterations used for Emax approximation; $i^* = r - I * int(\frac{r-1}{I})$ where $int(.)$ is an integer function that converts any real number to an integer by discarding its value after the decimal place.

2. Draw G^r_j (population mean of G_{ij}) from the posterior density (normal) computed by $\sigma^{r-1}_{G_j}$ and $\{G^{r-1}_{ij}\}^I_{i=1}$.

3. Draw $\sigma^r_{G_j}$ (population variance of G_{ij}) from the posterior density (inverse gamma) computed by G^r_j and $\{G^{r-1}_{ij}\}^I_{i=1}$.

4. For each i, use the Metropolis-Hastings algorithm to draw G^r_{ij}.

   (a) Use the prior on G_{ij} (i.e., $N(G^r_j, (\sigma^r_{G_j})^2)$) as the proposal distribution function to draw G^{*r}_{ij}.

(b) Compute the likelihood for consumer i at G_{ij}^{*r}, i.e., $\rho_i^r(G_{ij}^{*r}, \theta_1^{r-1})$ and determine whether or not to accept G_i^{*r}. The acceptance probability, λ, is given by

$$\begin{aligned} \lambda &= \min\left(\frac{\pi(G_{ij}^{*r}) \cdot \rho_i^r(G_{ij}^{*r}, \theta_1^{r-1}) \cdot q(G_{ij}^{*r}, G_{ij}^{r-1})}{\pi(G_{ij}^{r-1}) \cdot \rho_i^r(G_{ij}^{r-1}, \theta_1^{r-1}) \cdot q(G_{ij}^{r-1}, G_{ij}^{*r})}, 1\right) \\ &= \min\left(\frac{\rho_i^r(G_{ij}^{*r}, \theta_1^{r-1})}{\rho_i^r(G_{ij}^{r-1}, \theta_1^{r-1})}, 1\right) \end{aligned}$$

where the second equality follows from the fact that the proposal distribution is set to the prior on G_{ij}, i.e., $q(x, y) = \pi(y)$. Let G_i^r be the one that is accepted. Note that G_{ij}^{*r} will only affect $\hat{E}_p^r V(s, p; G_i^{*r}, \theta_1^{r-1})$ for consumer i and not for other consumers. Thus we only need to approximate the Emax functions for consumer i, using the weighted average of $\{E_p V^l(s, p; G_{i*}^l, \theta_1^{*l})\}_{l=r-N}^{r-1}$, treating G_i as one of the parameters when computing the weights. In the case of independent kernels, Emax approximation is,

$$\hat{E}_p^r V(s, p; G_i^{*r}, \theta_1^{r-1}) = \sum_{l=r-N}^{r-1} E_p \tilde{V}^l(s, p; G_{i*}^l, \theta_1^{r-1}) \frac{K_h(\theta_1^{r-1} - \theta_1^{*l}) K_h(G_i^{*r} - G_{i*}^l)}{\sum_{k=r-N}^{r-1} K_h(\theta_1^{r-1} - \theta_1^{*k}) K_h(G_i^{*r} - G_{i*}^k}$$

(c) Repeat for all i.

5. Use the Metropolis-Hastings algorithm to draw $\theta_1^r = (\alpha_1^r, \alpha_2^r, \gamma^r, \beta^r)$ conditional on G_{ij}^r.

(a) Draw $\theta_1^{*r} = (\alpha_1^{*r}, \alpha_2^{*r}, \gamma^{*r}, \beta^{*r})$ (candidate parameter vector).

(b) We then compute the likelihood conditional on $(\alpha_1^{*r}, \alpha_2^{*r}, \gamma^{*r}, \beta^{*r})$ and $\{G_i^r\}_{i=1}^I$, based on the approximated Emax functions, and determine whether or not to accept α_1^{*r}, α_2^{*r}, γ^{*r}, and β^{*r}. The Emax approximation is described as follows.

[7] Note that $\{K_h(\theta_1^{r-1} - \theta_1^{*l})\}_{l=r-N}^{r-1}$ is common across consumers. Therefore, one can calculate it outside the loop when programming this part.

For each i and s, $\hat{E}_p^r V(s,p;G_i^r,\theta_1^{*r})$ is obtained using the weighted average of the past value functions, $\{E_p\tilde{V}^l(s,p;G_{i^*}^l,\theta_1^{*l})\}_{l=r-N}^{r-1}$. In computing the weights for past value functions, we treat G_i as a parameter. Note that in the case of independent kernels, equation (6) becomes

$$\hat{E}_p^r V(s,p;G_i^r,\theta_1^{*r}) = \sum_{l=r-N}^{r-1} E_p\tilde{V}^l(s,p;G_{i^*}^l,\theta_1^{*l})\frac{K_h(\theta_1^{*r}-\theta_1^{*l})K_h(G_i^r-G_{i^*}^l)}{\sum_{k=r-N}^{r-1} K_h(\theta_1^{*r}-\theta_1^{*k})K_h(G_i^r-G_{i^*}^k)}.^{8}$$

6. Computation of pseudo-value function, $E_p\tilde{V}^r(s,p;G_{i^*}^r,\theta_1^{*r})$.

   (a) Make M draws of prices, p^m, from the price distribution.

   (b) Compute $\tilde{V}_0^r(s;\theta_1^{*r})$, $\tilde{V}_1^r(s,p_1^m;G_{i^*}^r,\theta_1^{*r})$ and $\tilde{V}_2^r(s,p_2^m;G_{i^*}^r,\theta_1^{*r})$, using the approximated Emax functions computed in Step 5 (c).

   (c) Given $\tilde{V}_0^r(s;\theta_1^{*r})$, $\tilde{V}_1^r(s,p_1^m;G_{i^*}^r,\theta_1^{*r})$ and $\tilde{V}_2^r(s,p_2^m;G_{i^*}^r,\theta_1^{*r})$, obtain the pseudo-value function, $\tilde{V}^r(s,p^m;G_{i^*}^r,\theta_1^{*r})$. By averaging $\tilde{V}^r(s,p^m;G_{i^*}^r,\theta_1^{*r})$ across p^m's, we integrate out prices and obtain $E_p\tilde{V}^r(s,p;G_{i^*}^r,\theta_1^{*r})$.

7. Compute the pseudo-likelihood, $\rho_i^{r+1}(G_i^r,\theta_1^r)$ $\forall i$, based on a updated set of past pseudo-value functions.

8. Go to iteration $r+1$.

In step 1 of the procedure described above, we pick one consumer in each iteration and store his/her pseudo-value function. Then, we use this pooled set of past pseudo-value functions across consumers to approximate the emax functions for all consumers. This is slightly different from IJC, who originally propose to store an individual-specific set of

[8]Note that $\{K_h(\theta_1^{*r}-\theta_1^{*l})\}_{l=r-N}^{r-1}$ is common across consumers. Therefore, one can compute it outside the loop that indexes consumers to save computational time.

past pseudo-value functions for each consumer. That is, in each iteration we store $H^r = \{\theta^{*l}, \{G_i^l, E_p\tilde{V}^l(.,p;G_i^l,\theta^{*l})\}_{i=1}^{I}\}_{l=r-N}^{r-1}$, and use $\{E_p\tilde{V}^l(.,p;G_i^l,\theta^{*l})\}_{l=r-N}^{r-1}$ to approximate consumer i's Emax function. One advantage of this approach is that the past pseudo-value functions used in the emax function approximation are more relevant to each consumer i, because they are evaluated at G_i^l's, which represent the posterior distribution of consumer i's value for the gift, and should be closer to G_i^{*r}. Note that this is not the case when we pool past pseudo-value functions across consumers because different consumers may have very different values of G_i. This suggests that if we store past pseudo-value functions individually, we may be able to reduce N in order to achieve the same level of precision for the emax approximation. This in turn should reduce the computation time. But one drawback is that we need much more memory to store past pseudo-value functions individually, although this may not be an important concern given the price of computer memory has been decreasing rapidly over time.

We only describe steps 2 and 3 briefly here. For the details of these two steps, we refer readers to Train (2003). When implementing step 5, it could be more efficient to separate them by blocks if the acceptance rate is low. The trade-off is that when implementing this step by blocks, we might also increase the number of expected future value approximation calculations and likelihood evaluations.

3.3 Choice of kernel's bandwidth and N

The IJC method relies on classical non-parametric methods to approximate the Emax functions using the past pseudo-value functions generated by the algorithm. One practical problem of nonparametric regression analysis is that the data becomes increasingly sparse as the dimensionality of the explanatory variables (x) increases. For instance, ten points that are uniformly distributed in the unit cube are more scattered than ten points distributed uniformly in the unit interval. Thus the number of observations available to provide information about the local behavior of an arbitrary regression function becomes small with large dimension. The curse of dimensionality of this non-parametric technique (in terms of number of parameters) could be something that we need to worry about.[9] The root of this problem is due to the bias-variance trade-off. In general, when the kernel bandwidth is small, the effective number of sample points available around x that influence the prediction would be small, making the prediction highly sensitive to that particular sample, i.e., yielding to high variance. When the kernel bandwidth is large, the prediction becomes overly smooth, i.e., yielding to high bias.

However, in implementing the IJC algorithm, the nature of this problem is different from the standard non-parametric estimation. Unlike a standard estimation problem where an econometrician cannot control the sample size of the data set, we can control the sample size for our nonparametric regressions by storing/using more past pseudo-value functions (i.e., increasing N). This is similar to the advantage of using the standard MCMC method to draw from the posterior distribution – the econometrician can control the number of iterations that requires to obtain convergence. Thus in practice, we expect that N may need to increase with the number of parameters in the model. As a result, it would also take more time to compute one iteration.

The discussion above suggests that the convergence rate is typically inversely related

[9]Note that this curse of dimensionality problem is different from that of solving for a dynamic programming model, where it refers to the size of the state space increasing exponentially with the number of state variables and the number of values for each state variable.

to the number of dimensions. But the situation that we face now is more subtle for two reasons. First, it is likely that the convergence rate is model specific, as the shape of the likelihood function is also model specific. Second, it should also depend on the data sample size. In general, when estimating a well-identified model with a data set with sufficient variation, the posterior variance of the parameters decreases with the sample size. This suggests that when the MCMC converges, the simulated parameter values would move within a small neighborhood of the posterior means. This implies that the set of past pseudo-value functions would be evaluated at parameter vectors that are concentrated in a small neighborhood. We expect that this should alleviate the curse of dimensionality problem.

It is worth discussing the impact of N on the estimation results. One implication is that as we increase N, older past pseudo-value functions will be used in the approximated Emax functions computation. This may result in slow improvements of the approximated emax values, and may slow down the speed of the MCMC convergence. As we decrease N, only more recent and accurate past pseudo-value functions will be used in the emax approximation. However, since the number of the past pseudo-value functions itself becomes smaller, the variance of the approximated emax values will increase. This may result in a higher standard deviation of the posterior distribution for some parameters. One way of mitigating this trade-off is to set N to be small at the beginning of the IJC algorithm and let N increase during the MCMC iterations. In this way, we can achieve a faster convergence and more stable posterior distributions at the same time. Another way to address this issue is to weight the past N pseudo-value functions differently so that the more recent pseudo-value functions receive higher weights (because they should be more accurate approximations). In one Monte Carlo experiment that we conduct in the next section, we show some evidence about the impact of N on the estimation results.

An obvious question that would likely come to researchers' mind is: How do we choose N and the bandwidth (h)? We believe that any suggested guidelines should ensure that the pseudo-value function gives us a good proxy for the true value function. We suggest that researchers check the distance between the pseudo-value function and the true value function during the estimation, and adjust N and h within the iterative process. For instance, researchers can store a large set of past pseudo-value functions (i.e., large N), and use the most recent $N' < N$ of them to do the approximation. This has the advantage that researchers can immediately increase N' if they discover that the approximation is not good enough. Researchers can start the algorithm with a small N', (say $N' = 100$), and an arbitrary bandwidth (say 0.01). Every 1000 iterations, they can compute the means of the MCMC draws, $\bar{\tilde{\theta}}$, and then compare the distance between the pseudo-value function and the exact value function at $\bar{\tilde{\theta}}$. If the distance is larger than what the researcher would accept, increase N'. Then use N' past pseudo-value functions to compute summary statistics and use standard optimal bandwidth formula (e.g., Silverman's rule of thumb; Silverman 1986, p.48) to set h. Of course, the cost of storing a large number of past pseudo-value function is that it requires more memory. But again thanks to the advance of computational power, the cost of memory is decreasing rapidly over time these days. Hence, we expect that memory would become less of a constraint in the near future. This suggestion would require us to solve for the DDP model exactly once every 1000 iterations. For complicated DDP models with random coefficients, this could still be computationally costly. But even in this case, one could simply compare the pseudo-value function and the exact value function at a small number of simulated heterogeneous parameter vectors, say 5. This would be equivalent to solving 5 homogeneous DDP models numerically and should be feasible even for complicated DDP models.

4 Estimation Results

To illustrate how to implement the IJC algorithm and investigate some of its properties, we conduct three Monte Carlo experiments. For each experiment, the simulated sample size is 1,000 consumers and 100 periods. We use the Gaussian kernel to weigh the past pseudo-value functions when approximating the Emax functions. The total number of MCMC iterations is 10,000, and we report the posterior distributions of parameters based on the 5,001-10,000 iterations. The sample size is 1,000 consumers for 100 periods. For all experiments, the following parameters are fixed: $\bar{S}_1, \bar{S}_2, \bar{p} = 1.0$, and $\sigma_p = 0.3$.

In the first experiment, we are interested in estimating a version of the model without unobserved heterogeneity. When simulating the data, we set $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\sigma_{G_1} = \sigma_{G_2} = \alpha_1 = \alpha_2 = 0$, $G_1 = 1.0$, $G_2 = 5.0$, $\gamma = -1.0$, and $\beta = 0.6$ or 0.8. Our goal is to estimate $\alpha_1, \alpha_2, G_1, G_2, \gamma$, and β, treating other parameters as known. To ensure that $\beta < 1$, we transform it as $\beta = \frac{1}{1+exp(\phi)}$ For all parameters, flat prior is used. Moreover, we use a random-walk proposal function. Table 1 summarizes the estimation results, and Figure 3 plots the simulated draws of parameters for the case of $\beta = 0.8$. The posterior means and standard deviations show that the IJC algorithm is able to recover the true parameter values well. Moreover, it appears that the MCMC draws converge after 2,000 iterations.

In the second experiment, we estimate a version of the model with unobserved heterogeneity. For simplicity, we only allow for consumer heterogeneity in G_2 (i.e., $\sigma_{G_1} = 0$).

The data is simulated based on the following parameter values: $\alpha_1 = \alpha_2 = 0.0$, $G_1 = 1.0$, $G_2 = 5.0$, $\sigma_{G_1} = 0.0$, $\sigma_{G_2} = 1.0$, γ, and $\beta = 0.6$ or 0.8. Again, we transform β by the logit formula, i.e., $\beta = \frac{1}{1+exp(\phi)}$. Our goal is to estimate $\alpha_1, \alpha_2, G_1, G_2, \sigma_{G_2}, \gamma$, and β, treating other parameters as known. For α_1, α_2, G_1, γ, and ϕ, we use flat prior. For G_2, we use a diffuse normal prior (i.e., setting the standard deviation of the prior to ∞). For σ_{G_2}, we use a diffuse inverted gamma prior, IG(ν_0, s_0) (i.e., setting $s_0 = 1$, $\nu_0 \rightarrow 1$). Table 2 shows the estimation results, and Figure 4 plots the simulated draws of parameters for $\beta = 0.8$. The IJC algorithm again is able to recover the true parameter values well. The MCMC draws appear to converge after 2,000 iterations for most of the parameters except G_1, which takes about 3,000 iterations to achieve convergence.

To learn more about the potential gain of IJC in terms of computational time, we compute the time per iteration and compare IJC's Bayesian MCMC algorithm with the full solution based Bayesian MCMC algorithm for both homogeneous model and heterogeneous model. In the full solution based Bayesian algorithm, we use 100 simulated draws of prices to integrate out the future price. For each model, we study three cases: $\beta = 0.6, 0.8$ and 0.98. Table 3 summarizes the results based on the average computation time. The computation time is based on a C program running in a linux workstation with Intel Core 2 Duo E4400 2GHz processor. Note that in the full solution based Bayesian algorithm, the computation time will increase as the discount factor becomes larger. This is because the number of steps required for convergence in a contraction mapping increases with the discount factor (i.e., the modulus). However, the computation time will not be influenced by the discount factor in the IJC algorithm. In the homogeneous model, the computation for the full solution based Bayesian is faster for $\beta = 0.6$ and 0.8. This is because (i) when β

is small, solving for a contraction mapping to get the exact Emax values is not that costly compared with computing the weighted emax values based on 1,000 past pseudo-value functions; (ii) full-solution based Bayesian approach does not need to perform step 5 in the homogeneous case, and step 7 in the heterogeneous case.[10] However, when $\beta = 0.98$, IJC algorithm is 40% faster than the full solution algorithm. In the heterogeneous model, we can see the advantage of the IJC algorithm much clearer. When $\beta = 0.6$, the IJC algorithm is 50% faster than the full solution based Bayesian algorithm; when $\beta = 0.8$, it is about 200% faster; when $\beta = 0.98$, it is about 3000% faster. In particular, it is clear that average computational time per iteration basically remains unchanged in the IJC algorithm. For the full solution based method, the computational time per iteration increases exponentially in β because, roughly speaking, we need to solve for the DDP model for each individual. If there are 1,000 individuals, the computational time will then be roughly (time per contraction mapping) X 1,000. For the heterogeneous model, with $\beta = 0.98$, it would take about 70 days to run the full solution based Bayesian MCMC algorithm for 10,000 iterations.[11] Using the IJC's Bayesian MCMC algorithm, it would take less than 2.5 days to obtain 10,000 iterations.

As discussed above, one issue in using the IJC is how to choose N, the number of the past pseudo-value functions. In the third Monte Carlo experiment, using the homogeneous model with $\beta = 0.98$, we investigate how changes in N influence the speed of convergence and the posterior distributions. Note that we use a high discount factor here. This is because, as we discussed in section 3.3, when the discount factor is large, the number

[10] In this exercise, we perform step 5 in the homogeneous case and step 7 in the heterogeneous case every time a candidate parameter vector is rejected.

[11] Depending on the convergence rate, the number of iterations required for Bayesian estimation could be higher than 10,000.

of past pseudo-value functions used for the Emax function approximation becomes more important. Thus, changes in N will have more impacts on the speed of convergence and the posterior distributions than when the discount factor is small. We simulate the data given the following set of parameter values: $\bar{S}_1 = 5$, $\bar{S}_2 = 10$, $\alpha_1 = \alpha_2 = 0$, $G_1 = 1$, $G_2 = 10$, $\gamma = -1$, $\bar{p} = 1.0$, and $\sigma_p = 0.3$. Our goal is to compare the performance of the IJC algorithm using $N = 100$ and $N = 1000$. Table 4 shows the posterior distributions of the parameters. The results show that the posterior means are very similar for both cases. But the standard deviations for G_1 and G_2 are smaller for $N = 1000$. This is consistent with our arguments earlier in section 3.3 – when using more pseudo-value functions to do the approximation, the variance of the approximation should become smaller. To see how the speed of convergence changes with N, we plot the MCMC samplers for α_1 and α_2 in Figure 5, and G_1 and G_2 in Figure 6. It can be seen that when $N = 100$, the speed of convergence is faster, but the paths also fluctuate more. Again, this is consistent with our discussion in section 3.3.

Note that when $\beta = 0.98$, the true parameter values are recovered less precisely, in particular, α_j and G_j. This is due to the identification problem that we discussed before – when the discount factor is large, it does not matter much when a consumer receives the gift. As a result, G_j would simply shift the choice probabilities, similar to the way that α_j does.

We now turn to discuss how to extend the IJC algorithm to (i) conduct policy experiments, and (ii) allow for continuous state space. We will also comment on the choice of kernels.

5 Extensions

5.1 Conducting Policy Experiments

The output of the IJC algorithm is posterior distribution for the parameters of the model, along with a set of value function (and emax function) estimates associated with each parameter vector. What if we are interested in a policy experiment that involves changing a policy parameter by certain percentage (e.g., increase the cost of entry by $100t$ percentage), such that the new parameter vectors do not belong to the support of the posterior distribution? It would appear that the IJC algorithm does not provide solutions of the dynamic programming problem evaluated at those policy parameter vectors. In fact, this limitation would apply even one uses full-solution based Bayesian MCMC algorithm.

Here we propose a minor modification of the IJC algorithm so that we can obtain the value functions of the new policy parameters as part of the estimation output as well. Suppose that the researcher is interested in the effect of changing θ_i to $\acute{\theta}_i$, where $\acute{\theta}_i = (1+t)\theta_i$ and θ_i is the ith element in the parameter vector θ. The modified procedure needs to store the following additional information: $\{\acute{\theta}^{*l}, E_p\tilde{V}^l(., p; \acute{\theta}^{*l})\}_{l=r-N}^{r-1}$, where $\acute{\theta}_i^{*l} = (1+t)\theta_i^{*l}$ and $\acute{\theta}_{-i}^{*l} = \theta_{-i}^{*l}$.

Once the MCMC samplers converge, we will have $\{\theta^l\}_{l=1}^L$ as well as $\{\acute{\theta}^{*l}, E_p\tilde{V}^l(., p; \acute{\theta}^{*l})\}_{l=L-N+1}^L$ as the outputs, where L is the total number of MCMC iterations. To do the policy experiment, (i) take the last M draws of θ^r, and set the draw of the policy parameter vector as follows: $\acute{\theta}_{-i}^r = \theta_{-i}^r$ and $\acute{\theta}_i^r = (1+t)\theta_i^r$; (ii) use $\{E_p\tilde{V}^l(., p; \acute{\theta}^{*l})\}_{l=L-N+1}^L$ to form an Emax at $\acute{\theta}^r$, and then obtain the value function and the choice probabilities at $\acute{\theta}^r$ for each r.

This procedure will increase the computational burden of each iteration due to the calculation of the approximated Emax at $\acute{\theta}^{*r}$. However, it is relatively straightforward to implement, and requires very little extra programming effort. To obtain some insights about how much more time it would take to include the results for a policy experiment, we break down the computation time of each iteration into four components based on our model with unobserved heterogeneity: (i) Emax approximation at θ^{*r}, (ii) likelihood evaluation at θ^{*r}, (iii) Emax approximation at θ^{r} based on an updated set of pseudo-value functions, (iv) likelihood evaluation at θ^{r} based on an updated set of pseudo-value functions. The results are shown in Table 5. Notice that to conduct the policy experiment, we only need to compute the Emax approximation at $\acute{\theta}^{*r}$, and store $\{E_p\tilde{V}^l(.,;p,\acute{\theta}^{*l})\}_{l=r-N}^{r-1}$. The steps used to calculate the Emax functions at $\acute{\theta}^{*r}$ are the same as those calculating the Emax functions at θ^{*r}. Thus the additional computational time will be the same as that for step (iii) above, which constitutes about 40-60% of the computation time per iteration. This indicates that it would roughly increase the computational time by 40-60% if we use the IJC algorithm to conduct a policy experiment as well.

Finally, we note that there is a limitation of this procedure: we need to know the magnitude of the change in the policy parameter before seeing the estimation results. Sometimes researchers may not be able to determine this until they obtain the parameter estimates.

5.2 Continuous State Space

The state space of the dynamic store choice model described earlier is the number of stamps collected, which takes a finite number of values. In many marketing and economics applications, however, we have to deal with continuous state variables such as prices, advertising expenditures, capital stocks, etc. IJC also describes how to extend the algorithm to combine with the random grid approximation proposed by Rust (1997). To illustrate how it works, we consider the homogeneous model here.

Consider a modified version of the dynamic store choice model without unobserved consumer heterogeneity. Suppose that prices set by the two supermarkets follow a first-order Markov process (instead of an iid process across time): $f(p'|p;\theta_p)$, where θ_p is the vector of parameters for the price process. In this setting, the expected value functions in equation (5) are conditional on current prices, $E_{p'}[V(s',p';\theta)|p]$. In the Rust random grid approximation, we evaluate this expected value function as follows. We randomly sample M grid points, $p^m = (p_1^m, p_2^m)$ for $m = 1,\ldots,M$. Then we evaluate the value functions at each p^m and compute the weighted average of the value functions, where weights are given by the conditional price distribution.

For each iteration r, we can make one draw of prices, $p^r = (p_1^r, p_2^r)$, from a distribution. For example, we can define this distribution as uniform on $[\underline{p},\bar{p}]^2$ where $\underline{p}$ and $\bar{p}$ are the lowest and highest observed prices, respectively. Then, we compute the pseudo-value function at p^r, $\tilde{V}^r(s,p^r;\theta^r)$ for all s. Thus, H^r in step 1 of section 3.2.1 needs to be changed to

$$H^r = \{\{\theta^{*l}, p^l, \tilde{V}^l(.,p^l;\theta^{*l})\}_{l=r-N}^{r-1}, \rho^{r-1}(\theta^{r-1})\}.$$

The expected value function given s', p, and θ^r is then approximated as follows.

$$\hat{E}_{p'}^r[V(s',p';\theta^r)|p] = \sum_{l=r-N}^{r-1} \tilde{V}^l(s',p^l;\theta^{*l}) \frac{K_h(\theta^r-\theta^{*l}) f(p^l|p;\theta_p)}{\sum_{k=r-N}^{r-1} K_h(\theta^r-\theta^{*k}) f(p^k|p;\theta_p)}. \tag{8}$$

Unlike the Rust random grid approximation which fix the number of grid points throughout the estimation, the random grid points her change at each MCMC iteration. In addition, the total number of random grid points can be made arbitrarily large by increasing N.

The procedure for obtaining the pseudo-value function in step 4 of section 3.2.1 needs to be modified slightly. We store $\tilde{V}^r(s, p^r; \theta^{*r})$ instead of $E_p\tilde{V}^r(s, p; \theta^{*r})$. Specifically, the pseudo-value function at p^r (and θ^{*r}) is computed as follows. For each s,

$$\tilde{V}^r(s, p^r; \theta^{*r}) = \log(\exp(\tilde{V}_0^r(s; \theta^{*r})) + \exp(\tilde{V}_1^r(s, p_1^r; \theta^{*r})) + \exp(\tilde{V}_2^r(s, p_2^r; \theta^{*r}))),$$

where

$$\tilde{V}_j^r(s, p_j^r; \theta^{*r}) = \begin{cases} \alpha_j - \gamma p_j^r + G_j + \beta \hat{E}_{p'}^r[V(s', p'; \theta^{*r})|p^r] & \text{if } s_j = \bar{S}_j - 1, \\ \alpha_j - \gamma p_j^r + \beta \hat{E}_{p'}^r[V(s', p'; \theta^{*r})|p^r] & \text{otherwise,} \end{cases}$$
$$\tilde{V}_0^r(s; \theta^{*r}) = \beta \hat{E}_{p'}^r[V(s', p'; \theta^{*r})|p^r].$$

The approximated Emax functions above are computed by equation (8).

Note that if we simply apply the Rust random grid approximation with M grid points in the IJC algorithm, we need to compute the pseudo-value functions at M grid points in each iteration. Also, the integration with respect to prices requires us to first compute the approximated value function at each grid point and then take the weighted average of the approximated value functions. The computational advantage of the IJC random grid algorithm described above comes from the fact that we only need to compute the pseudo-value function at one grid point, p^r, in each iteration, and the integration with respect to prices can be done without approximating the value functions at a set of grid points separately.

5.3 Choice of Kernels

It should be noted that there are many kernels that one could use in forming a non-parametric approximation for the Emax function. IJC discuss their method in terms of the Gaussian kernel. Norets (2008) extends IJC's method by approximating the emax function using the past value functions evaluated at the "nearest neighbors," and allowing the error terms to be serially correlated. At this point, the relative performances of different kernels in this setting are still largely unknown. It is possible that for models with certain features, the Gaussian kernel performs better than other kernels in approximating the pseudo-value function, while other kernels may outperform the Gaussian kernel for models with other features. More research is needed to document the pros and cons of different kernels, and provide guidance in the choice of kernel when implementing the IJC method.

6 Conclusion

In this paper, we discuss how to implement the IJC method using a dynamic store choice model. For illustration purpose, the specification of the model is relatively simple. We believe that this new method is quite promising in estimating DDP models. Osborne (2007) has successfully applied this method to estimate a much more detailed consumer learning model. The IJC method allows him to incorporate much more general unobserved consumer heterogeneity than the previous literature, and draw inference on the relative importance of switching costs, consumer learning and consumer heterogeneity in explaining customers persistent purchase behavior observed in scanner panel data. Ching et al. (2009) have also successfully estimated a learning and forgetting model where consumers are forward-looking.

Bayesian inference has allowed researchers and practitioners to develop more realistic static choice models in the last two decades. It is our hope that the new method presented here and its extensions would allow us to take another step to develop more realistic dynamic choice models and ease the burden of estimating them in the near future.

References

Ackerberg, Daniel A. 2001. A New Use of Importance Sampling to Reduce Computational Burden in Simulation Estimation. Working paper, Department of Economics, UCLA.

Ackerberg, Daniel A. 2003. Advertising, Learning, and Consumer Choice in Experience Good Markets: An Empirical Examination. *International Economic Review* **44**(3) 1007–1040.

Aguirreagabiria, Victor, Pedro Mira. 2002. Swapping the Nested Fixed Point Algorithm: A Class of Estimators for Discrete Markov Decision Models. *Econometrica* **70**(4) 1519–1543.

Albert, James H., Siddhartha Chib. 1993. Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association* **88** 669–679.

Allenby, Greg M. 1994. An Introduction to Hierarchical Bayesian Modeling. Tutorial Notes, Advanced Research Techniques Forum, American Marketing Association.

Allenby, Greg M., Peter J. Lenk. 1994. Modeling Household Purchase Behavior with Logistic Normal Regression. *Journal of the American Statistical Association* **89** 1218–1231.

Brown, Meta, Christopher J. Flinn. 2006. Investment in Child Quality Over Marital States. Working paper, Department of Economics, New York University.

Černý, V. 1985. Thermodynamical Approach to the Travelling Salesman Problem: An

Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications* **45**(1) 41–51.

Crawford, Gregory S., Matthew Shum. 2005. Uncertainty and Learning in Pharmaceutical Demand. *Econometrica* **73**(4) 1137–1174.

Diermeier, Daniel, Michael P. Keane, Antonio M. Merlo. 2005. A Political Economy Model of Congressional Careers. *American Economic Review* **95** 347–373.

Erdem, Tülin, Susumu Imai, Michael P. Keane. 2003. Brand and Quality Choice Dynamics under Price Uncertainty. *Quantitative Marketing and Economics* **1**(1) 5–64.

Erdem, Tülin, Michael P. Keane. 1996. Decision Making under Uncertainty: Capturing Dynamic Brand Choice Processes in Turbulent Consumer Goods Markets. *Marketing Science* **15**(1) 1–20.

Geweke, John F., Michael P. Keane. 2002. Bayesian Inference for Dynamic Discrete Choice Models Without the Need for Dynamic Programming. Mariano, Schuermann, Weeks, eds., *Simulation Based Inference and Econometrics: Methods and Applications*. Cambridge University Press, Cambridge, UK.

Gönül, Füsun, Kannan Srinivasan. 1996. Estimating the Impact of Consumer Expectations of Coupons on Purchase Behavior: A Dynamic Structural Model. *Marketing Science* **15**(3) 262–279.

Hartmann, Wesley R., V. Brian Viard. 2008. Do Frequency Reward Programs Create Switching Costs? A Dynamic Structural Analysis of Demand in a Reward Program. *Quantitative Marketing and Economics* **6**(2) 109–137.

Hendel, Igal, Aviv Nevo. 2006. Measuring the Implications of Sales and Consumer Inventory Behavior. *Econometrica* **74**(6) 1637–1673.

Hitsch, Günter. 2006. An Empirical Model of Optimal Dynamic Product Launch and Exit Under Demand Uncertainty. *Marketing Science* **25**(1) 25–50.

Hotz, Joseph V., Robert Miller. 1993. Conditional Choice Probabilities and the Estimation of Dynamic Models. *Review of Economic Studies* **60**(3) 497–529.

Imai, Susumu, Neelam Jain, Andrew Ching. 2008. Bayesian Estimation of Dynamic Discrete Choice Models. Conditionally accepted at *Econometrica*. Available at SSRN: http://ssrn.com/abstract=1118130.

Imai, Susumu, Kala Krishna. 2004. Employment, Deterrence and Crime in a Dynamic Model. *International Economic Review* **45**(3) 845–872.

Keane, Michael P., Kenneth I. Wolpin. 1994. The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence. *Review of Economics and Statistics* **74**(4) 648–672.

Keane, Michael P., Kenneth I. Wolpin. 1997. The Career Decisions of Young Men. *Journal of Political Economy* **105** 473–521.

Kirkpatrick, S., C.D. Gelatt, M.P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* **220** 671–680.

Lancaster, Tony. 1997. Exact Structural Inference in Optimal Job Search Models. *Journal of Business and Economic Statistics* **15**(2) 165–179.

McCulloch, Robert, Peter E. Rossi. 1994. An Exact Likelihood Analysis of the Multinomial Probit Model. *Journal of Econometrics* **64** 207–240.

Norets, Andriy. 2008. Inference in Dynamic Discrete Choice Models with Serially Correlated Unobserved State Variables.

Osborne, Matthew. 2007. Consumer Learning, Switching Costs, and Heterogeneity: A Structural Examination. Working paper, U.S. Department of Justice.

Rossi, Peter E., Greg M. Allenby. 1999. Marketing Models of Consumer Heterogeneity. *Journal of Econometrics* **89** 57–78.

Rossi, Peter E., Robert McCulloch, Greg M. Allenby. 1996. The Value of Purchase History Data in Target Marketing. *Marketing Science* **15** 321–340.

Rust, John. 1987. Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher. *Econometrica* **55**(5) 999–1033.

Rust, John. 1997. Using Randomization to Break the Curse of Dimensionality. *Econometrica* **65**(3) 487–516.

Silverman, Bernard W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, UK.

Song, Inseong, Pradeep K. Chintagunta. 2003. A Micromodel of New Product Adoption with Heterogeneous and Forward Looking Consumers: Application to the Digital Camera Category. *Quantitative Marketing and Economics* **1**(4) 371–407.

Sun, Baohong. 2005. Promotion Effect on Endogenous Consumption. *Marketing Science* **24**(3) 430–443.

Train, Kenneth E. 2003. *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge, UK.

Yang, Botao, Andrew Ching. 2008. Dynamics of Consumer Adoption Decisions of Financial Innovation: The Case of ATM Cards in Italy. Working paper, Rotman School of Management, University of Toronto.

Table 1: Estimation Results: Homogeneous Model

		$\beta = 0.6$		$\beta = 0.8$	
parameter	TRUE	mean	sd	mean	sd
α_1 (intercept for store 1)	0.0	-0.001	0.019	-0.030	0.022
α_2 (intercept for store 2)	0.0	-0.002	0.019	-0.018	0.028
G_1 (reward for store 1)	1.0	0.998	0.017	1.052	0.021
G_2 (reward for store 2)	5.0	5.032	0.048	5.088	0.085
γ (price coefficient)	-1.0	-0.999	0.016	-0.996	0.019
β (discount factor)	0.6/0.8	0.601	0.008	0.800	0.010

Notes
Sample size: 1,000 consumers for 100 periods.
Fixed parameters: $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\bar{p} = 1.0$, $\sigma_p = 0.3$, $\sigma_{G_j} = 0$ for $j = 1, 2$.
Turning parameters: $N = 1,000$ (number of past pseudo-value functions used for emax approximations), $h = 0.01$ (bandwidth).

Table 2: Estimation Results: Heterogeneous Model

		$\beta = 0.6$		$\beta = 0.8$	
parameter	TRUE	mean	sd	mean	sd
α_1 (intercept for store 1)	0.0	-0.005	0.019	-0.022	0.022
α_2 (intercept for store 2)	0.0	0.010	0.021	0.005	0.037
G_1 (reward for store 1)	1.0	1.017	0.017	1.010	0.019
G_2 (reward for store 2)	5.0	5.066	0.065	4.945	0.130
σ_{G2} (sd of G_2)	1.0	1.034	0.046	1.029	0.040
γ (price coefficient)	-1.0	-1.004	0.016	-0.985	0.019
β (discount factor)	0.6/0.8	0.595	0.005	0.798	0.006

Notes
Sample size: 1,000 consumers for 100 periods.
Fixed parameters: $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\bar{p} = 1.0$, $\sigma_p = 0.3$, $\sigma_{G_1} = 0$.
Turning parameters: $N = 1,000$ (number of past pseudo-value functions used for emax approximations), $h = 0.01$ (bandwidth).

Table 3: Computation Time Per MCMC Iteration (in seconds)

	Homogeneous Model			Heterogeneous Model		
algorithm	$\beta = 0.6$	$\beta = 0.8$	$\beta = 0.98$	$\beta = 0.6$	$\beta = 0.8$	$\beta = 0.98$
Full solution based Bayesian	0.782	0.807	1.410	31.526	65.380	613.26
IJC with N=1000	1.071	1.049	1.006	19.300	19.599	18.387

Notes
Sample size: 1,000 consumers for 100 periods.
Number of state points: 8 ($\bar{S}_1 = 2$, $\bar{S}_2 = 4$).
Number of parameters: 6 in homogeneous model; 7 in heterogeneous model.

Table 4: The Impact of N

		N=100		N=1000	
parameter	TRUE	mean	sd	mean	sd
α_1 (intercept for store 1)	0.0	-0.049	0.020	-0.061	0.020
α_2 (intercept for store 2)	0.0	0.032	0.019	0.022	0.019
G_1 (reward for store 1)	1.0	1.234	0.034	1.246	0.021
G_2 (reward for store 2)	10.0	9.740	0.063	9.751	0.028
γ (price coefficient)	-1.0	-1.000	0.018	-0.991	0.018

Notes
Sample size: 1,000 consumers for 100 periods.
Fixed parameters: $\bar{S}_1 = 5$, $\bar{S}_2 = 10$, $\bar{p} = 1.0$, $\sigma_p = 0.3$, $\sigma_{G_j} = 0$ for $j = 1, 2$, $\beta = 0.98$.
Turning parameters: $h = 0.01$ (bandwidth).

Table 5: Breakdown of Computation Time Per MCMC Iteration (in seconds) for IJC Algorithm ($\beta = 0.8$)

	Heterogeneous Model		
computation	N=100	N=500	N=1000
Emax approximation at θ^{*r} (steps 4(b) & 5(b))	1.1502	5.6808	11.3149
Likelihood value at θ^{*r} (steps 4(b) & 5(b))	0.5229	0.5174	0.5264
Emax approximation at θ^{r} (step 7)	0.7180	3.5097	6.9819
Likelihood value at θ^{r} (step 7)	0.3280	0.3223	0.3271
computation time per iteration	2.7724	10.1219	19.2253

Notes
Sample size: 1,000 consumers for 100 periods.
Number of state points: 8 ($\bar{S}_1 = 2$, $\bar{S}_2 = 4$).
Number of parameters: 7.
Steps indicated in the table are in section 3.2.2.
Step 7 was performed every time a candidate parameter value was rejected.

Figure 1: Choice probabilities across states for different discount factors

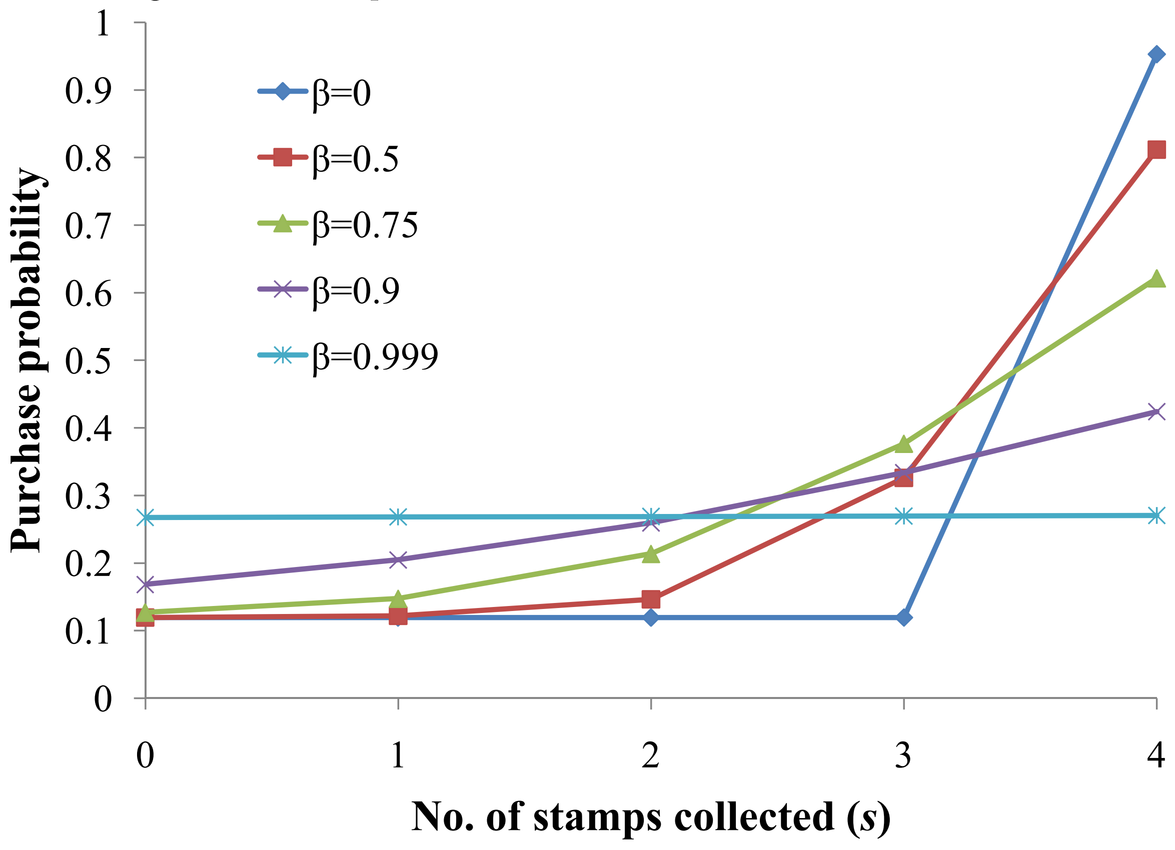


Figure 2: Choice probabilities for different discount factors across states

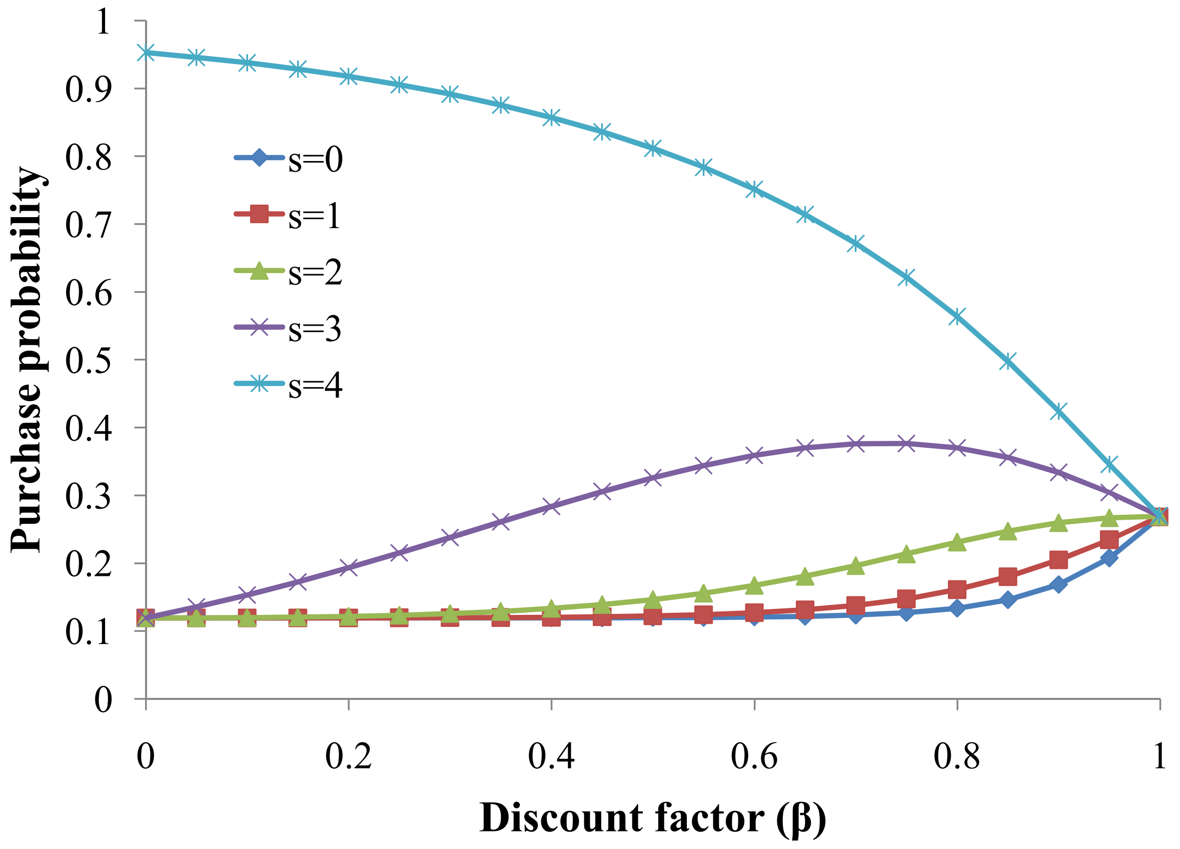
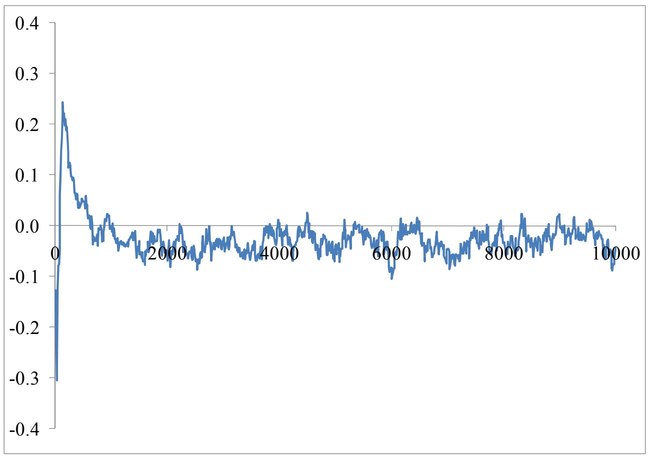
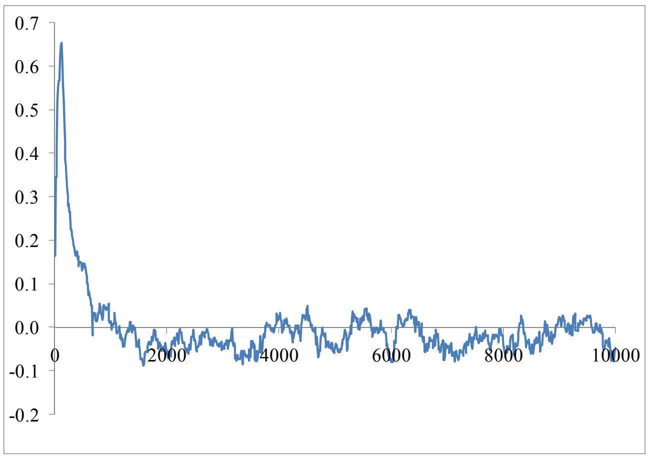


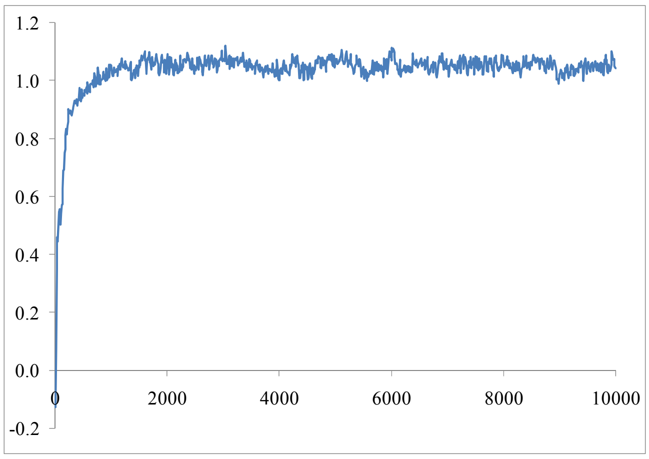
Figure 3: MCMC plots: Homogeneous Model with $\beta = 0.8$



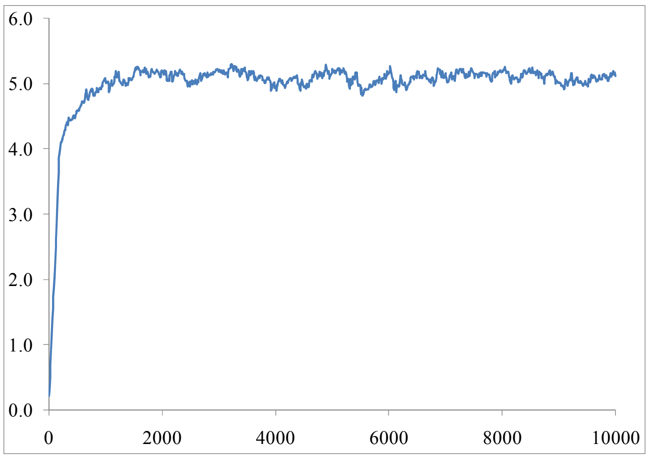
α_1 (true value = 0.0)



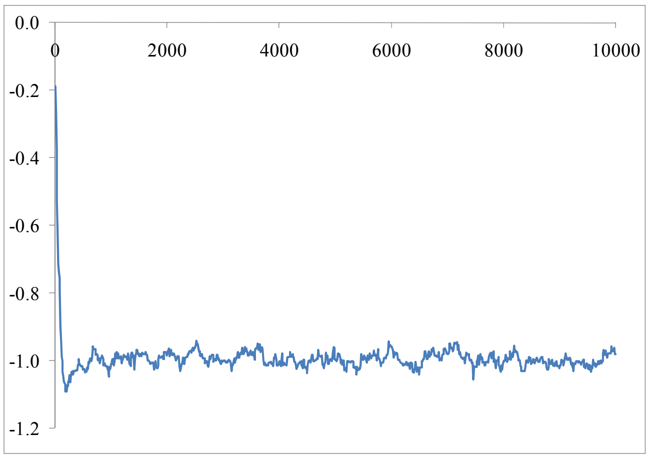
α_2 (true value = 0.0)



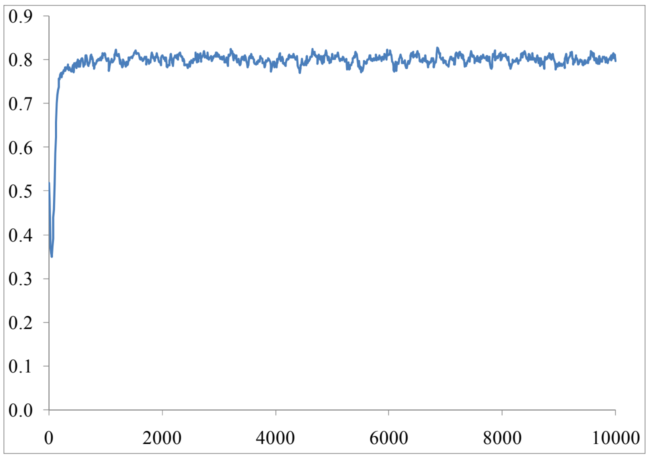
G_1 (true value = 1.0)



G_2 (true value = 5.0)

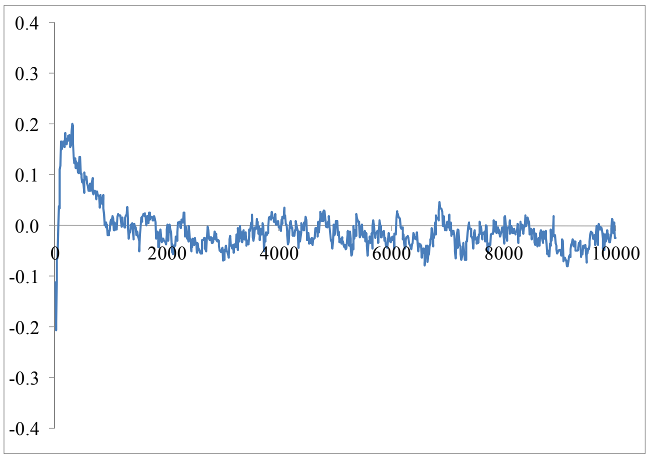


γ (true value = -1.0)

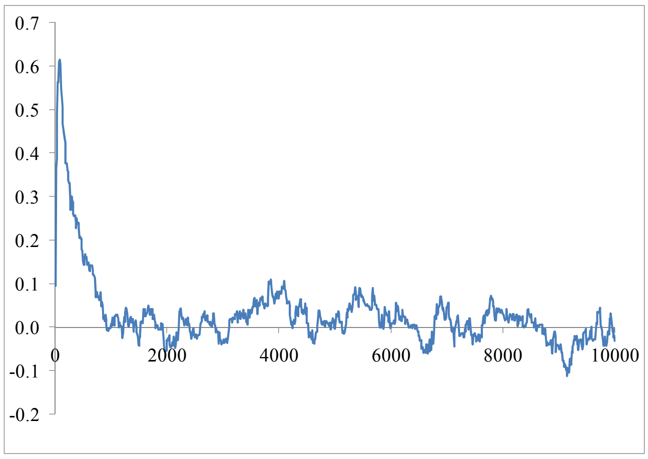


β (true value = 0.8)

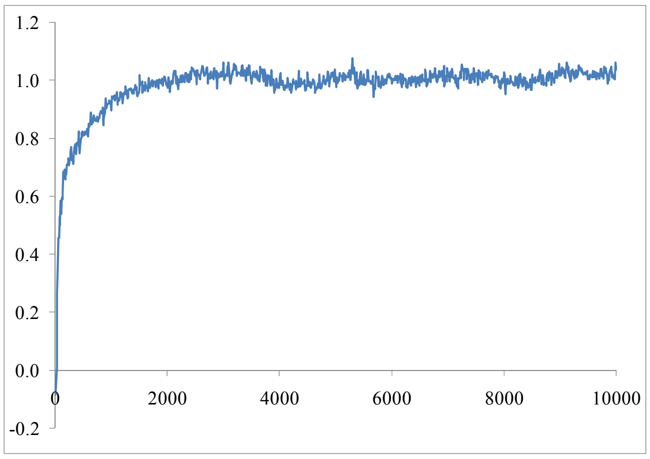
Figure 4: MCMC plots: Heterogeneous Model with $\beta = 0.8$



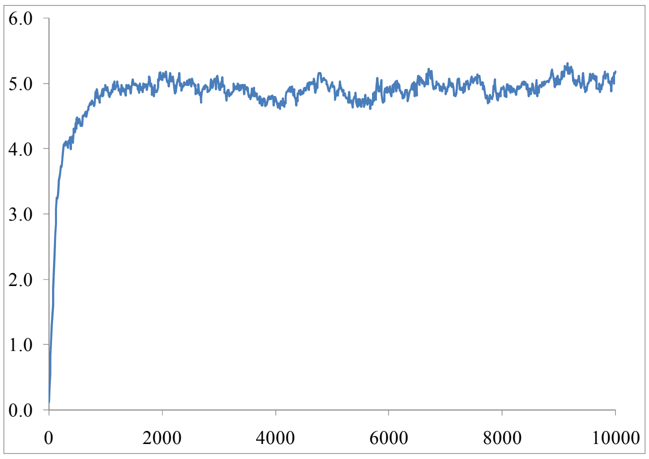
α_1 (true value = 0.0)



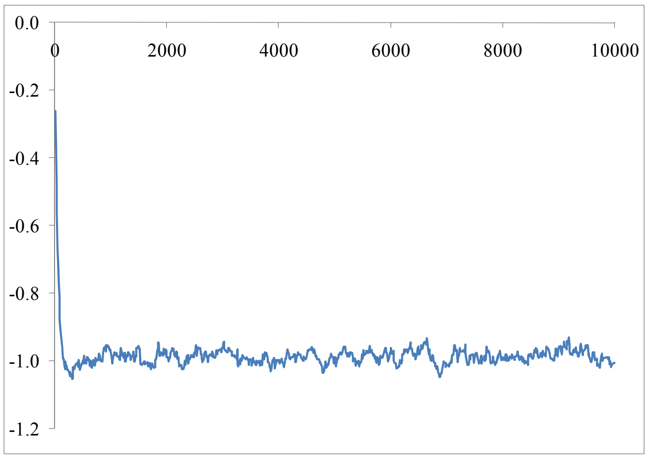
α_2 (true value = 0.0)



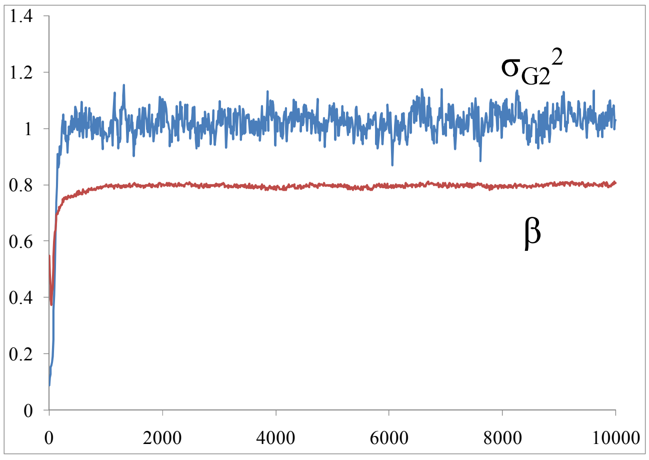
G_1 (true value = 1.0)



G_2 (true value = 5.0)



γ (true value = -1.0)



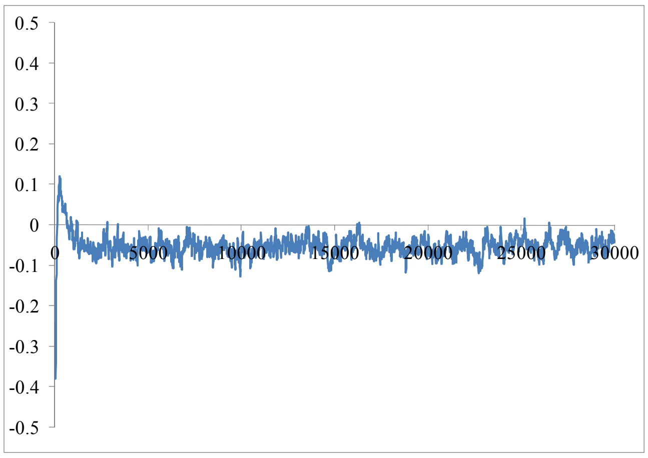
β (true value = 0.8)

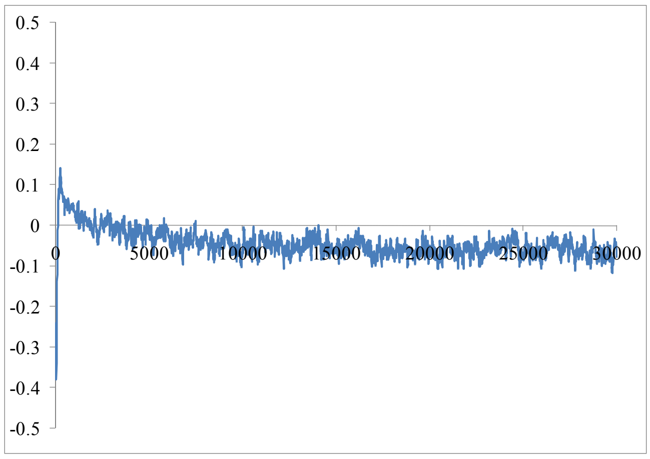
σ_{G2}^2 (true value = 1.0)

Figure 5: MCMC plots: Impact of N on α_1 and α_2 when $\beta = 0.98$

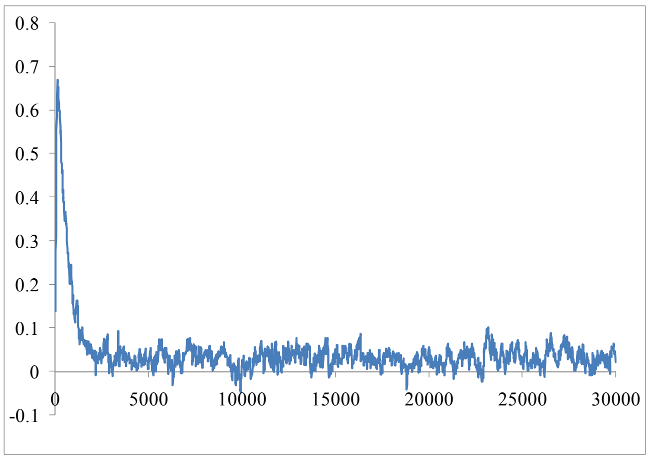
N = 100 N = 1000

α_1 (true value = 0.0)





α_2 (true value = 0.0)



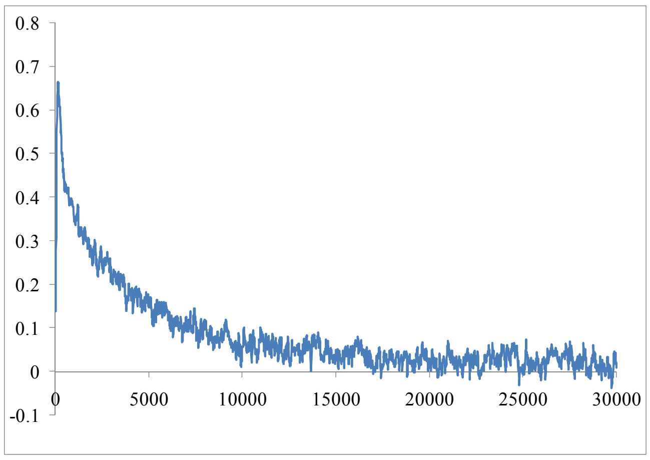
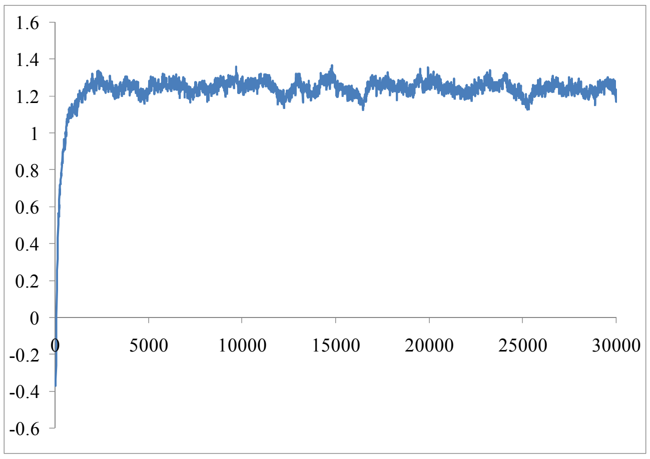
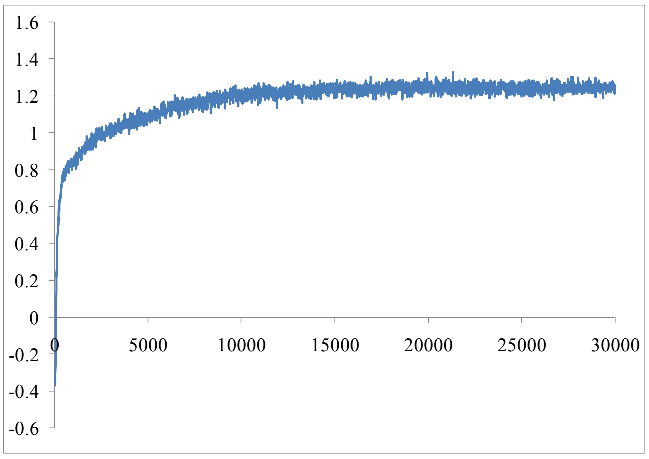


Figure 6: MCMC plots: Impact of N on G_1 and G_2 when $\beta = 0.98$

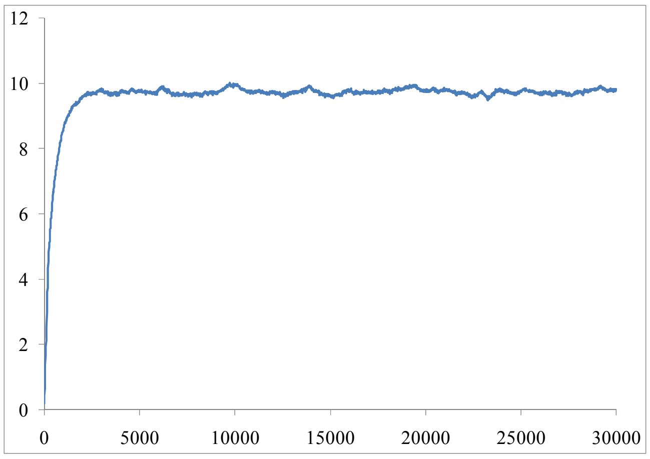
N = 100 N = 1000

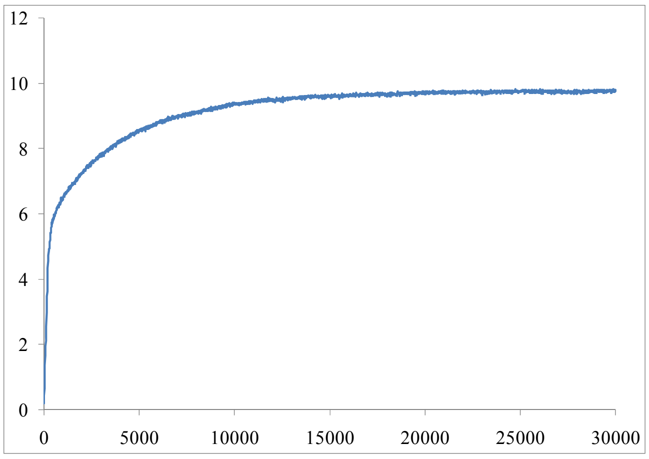
G_1 (true value = 1.0)





G_2 (true value = 5.0)


12
10
8
6
4
2
0
0
5000
10000
15000
20000
25000
30000


12
10
8
6
4
2
0
0
5000
10000
15000
20000
25000
30000