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: March 11, 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 (Rust 1987), career decision choice (Keane and Wolpin 1997; Diermier, Merlo and Keane 2005), choice to commit crimes (Imai and Krishna 2004), dynamic brand choice (Erdem and Keane 1996; Gönül and Srinivasan 1996; Sun 2005), dynamic quantity choice (Erdem, Imai and Keane 2003; 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). Neverthe-

less, 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 Bayesian MCMC typically requires a lot more iterations than classical approach to get 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 et al. (2008) (IJC) propose a new modified MCMC algorithm 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 needs 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. Section 4 shows the estimation results using the IJC method. Section 5 discusses how to extend this method to conduct policy experiments and incorporate continuous state variables. 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. Let $s_{jt} \in \{0, 1, \ldots, \bar{S}_j - 1\}$ denote the number of stamps collected for store j in period t before consumers make a decision. Note that s_{jt} 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 $s_t = (s_{1t}, s_{2t})$ is given by

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

where α_j is the loyalty 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., $U_{i0t} = \epsilon_{i0t}$.

The 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 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_{jt}, the next period state, s_{jt+1}, is determined as follows:

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

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, the Bellman's equation for consumer i is given by

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

where the second equality follows from the extreme value assumption on ϵ. The alternative-specific value functions are written as

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

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

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

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

We assume that prices of store j in each period are drawn from an iid normal distribution, $N(\bar{p}, \sigma_p^2)$. Also, we assume that this price distribution is known to consumers.

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

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), and two exogenous state variables (p_{i1t}, p_{i2t}), 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 Intertemporal trade-off and the discount factor

The main dynamics of the model is the intertemporal trade-off created by the reward program. Suppose that a consumer is close to 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 discount factor is less than one, the delay will lower the present discounted value of the reward.

Thus he or she may have more incentive to choose store 1 today. This incentive will increase as he or she is closer to the completion of the stamp card. Note that when the discount factor is one, when to receive a reward does not matter for consumers. Thus the intertemporal trade-off becomes irrelevant.

The dynamics illustrated above suggests that the empirical choice frequency of visiting the stores across states could 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 current 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 ($\beta = 0.999$). In this case, consumers hardly discount the future utilities. Thus, the timing of receiving the reward is irrelevant for their decision today. As shown in the figure, the choice probability is flat when $\beta = 0.999$. As β decreases from one, the choice probability decreases for smaller s and increases for larger s. The former happens because when the number of stamps collected now is small, the completion date of the stamp card is still very distant. Thus, as β decreases, the incentive to collect a stamp today decreases, resulting in a lower choice probability. For the latter, note that when $\beta < 1$, the closer consumers are to the completion of the stamp card, the larger the loss from delaying the completion. This loss is further amplified as β decreases.[1]

[1]Hartmann and Viard (2008) also discussed how the discount factor would affect the pattern of choice

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. 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. We will discuss how to obtain V^{l+1}.

2. For each j, we make M draws of p_j^m from the price distribution function, $N(\bar{p}, \sigma_p^2)$.

3. Substitute $\{p^m\}$ into $V^l(s; p, \theta)$, and then take the average across $\{p^m\}$ 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 operators and obtain $V^{l+1}(s; p, \theta), \forall s, p$.

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

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 until it converges.

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

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).

value functions will need to be solved in each set of trial parameter values (for maximum likelihood), or each set of random draw of parameter values (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. 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 propose 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 proposes to replace the contraction mapping procedure of solving the value functions with a weighted average 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 could

be computationally much cheaper than the standard contraction mapping procedure. Combining these two insights, IJC dramatically reduce the computational burden of each estimation iteration. 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, $H^r = \{\theta^l, E_p\tilde{V}^l(s;p,\theta^l)\}_{l=r-N}^{r-1}$. Let's extend Bellman Equations (1)-(2) 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,

$$\tilde{V}^r(s;p,\theta^r) \;=\; log(exp(\tilde{V}_1^r(s;p_1,\theta^r)) + exp(\tilde{V}_2^r(s;p_2,\theta^r)) + exp(\tilde{V}_0^r(s;\theta^r))), \qquad (3)$$

where

$$\begin{aligned}
\tilde{V}_j^r(s;p_j,\theta^r) &= \begin{cases} \alpha_j - \gamma p_j + G_j + \beta\hat{E}_{p'}^r V(s';p',\theta^r) & \text{if } s_j = N_j - 1, \\ \alpha_j - \gamma p_j + \beta\hat{E}_{p'}^r V(s';p',\theta^r) & \text{otherwise,} \end{cases} \\
\tilde{V}_0^r(s;\theta^r) &= \beta\hat{E}_{p'}^r V(s';p',\theta^r).
\end{aligned} \qquad (4)$$

Note that $s_j' = 0$ if $s_j = N_j - 1$, and $s_j' = s_j + 1$ otherwise, and $s_{-j}' = s_{-j}$ (see equation (1)).

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:

$$\hat{E}_p^r V(s;p,\theta^r) = \sum_{n=1}^{N} E_p \tilde{V}^{r-n}(s;p,\theta^{r-n}) \frac{K_h(\theta^r - \theta^{r-n})}{\sum_{k=1}^{N} K_h(\theta^r - \theta^{t-k})}, \tag{5}$$

where $K_h(.)$ is a gaussian kernel with bandwidth h. To obtain $E_p\tilde{V}^r(s;p,\theta^r)$, we could simulate a set of p's, and evaluate $\tilde{V}_j^r$ at those simulated prices. But in our model here, we assume prices are observed. Therefore, one can simply evaluate $\tilde{V}_j^r$ at the observed prices, and average $\tilde{V}^r(s;p^d,\theta^r)$ across p^d's to obtain $E_p\tilde{V}^r(s;p,\theta^r)$.

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 equation (5) is the key innovation of IJC. In principles, this step is also applicable in classical estimation methods such as GMM and Maximum Likelihood. 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. However, there are at least several advantages of implementing IJC's pseudo-value function approach in Bayesian estimation. First, the non-parametric approximation in equation (5) would be more efficient if the past pseudo-value functions are evaluated at θ^{r-n}'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. Second, in the presence of unobserved consumer heterogeneity (or in general complicated choice models), 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. Even for static choice problems, it is common that the likelihood function is multi-modal when unobserved heterogeneity is allowed. It is likely that the likelihood for DDP models with unobserved heterogeneity would be very complicated too.

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 perform better for models with different 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.

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 illustrate the implementation of the IJC algorithm by using the dynamic store choice model described above. Let $I_{buy,ijt}$ be an indicator function for purchasing at store j by consumer i at time t, and $p_{it} = (p_{ijt}, p_{ijt})$ 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 IJC. 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 without too many difficulties.

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 $\theta = (\alpha_1, \alpha_2, G_1, G_2, \gamma, \beta)$. We fix the rest of the parameters: $\bar{p}$, and σ_p.

The IJC algorithm generates a sequence of MCMC draw of $\theta^r, r = 1, 2, ...$ The algorithm modifies the Metropolis-Hastings within Gibbs algorithm, and involves drawing a candidate parameter θ^{*r} in each iteration. 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 $\{\theta^l, E_p\tilde{V}^l(s; p, \theta^l)\}_{l=r-N}^{r-1}$. The reason is that the acceptance rate of the M-H step could be low. If we choose to store $\{\theta^l, E_p\tilde{V}^l(s; p, \theta^l)\}_{l=r-N}^{r-1}$, there may be a significant portion of θ^l's that are repeated. Nevertheless, 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, $\hat{E}_p^r V(.;.,\theta^{*r})$, to form the likelihood. As we will show below, it will only take an extra step to obtain $E_p\tilde{V}^r(.;.,\theta^{*r})$. So storing $\{E_p\tilde{V}^r(.;.,\theta^{*l})\}_{l=r-N}^{r-1}$ will impose little extra costs.

Now we turn to discuss the details of the implementation. For each iteration r,

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

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

   where N is the number of past iterations used for Emax approximation; $E_p\tilde{V}^l(s;,.)$ is the expected pseudo-value functions w.r.t. p; $\rho^{r-1}(\theta^{r-1})$ is the pseudo-likelihood of the data conditional on θ^{r-1} at iteration $r-1$. Note that we store the expected pseudo-value functions w.r.t. price (i.e., $E_p\tilde{V}^l$).

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

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, $min(\frac{\pi(\theta^{*r}).\rho^r(\theta^{*r}).q(\theta^{*r},\theta^{r-1})}{\pi(\theta^{r-1}).\rho^{r-1}(\theta^{r-1}).q(\theta^{r-1},\theta^{*r})},1)$. 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 $\theta^r = \theta^{r-1}$. The Emax approximation is as follows:

   (a) For each s, $\hat{E}_p^r V(s;a,\theta^{*r})$ is obtained using the weighted average of the past expected pseudo-value functions: $\{E_p\tilde{V}^l(s;p,\theta^{*l})\}_{l=r-N}^{r-1}$. The weight of each

past expected pseudo-value function is determined by gaussian kernels.

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

(b) Compute $\tilde{V}_0^r(s;\theta^{*r})$, $\tilde{V}_1^r(s;p_{i1t}^d,\theta^{*r})$ and $\tilde{V}_2^r(s;p_{i2t}^d,\theta^{*r})$, by equation (4), using the approximated Emax functions computed above, where p_{ijt}^d is the observed price for consumer i at store j in period t. These alternative specific pseudo-value functions will be used to construct the pseudo-likelihood of θ^{*r}.

4. Compute the pseudo-likelihood, $\rho^r(\theta^r)$. Note that we will only need to redo this computation if we reject θ^{*r}.

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

   (a) Given $\tilde{V}_0^r(s;\theta^{*r})$, $\tilde{V}_1^r(s;p_{i1t}^d,\theta^{*r})$ and $\tilde{V}_2^r(s;p_{i2t}^d,\theta^{*r})$, we can obtain the pseudo-value function, $\tilde{V}^r(s;p_{it}^d,\theta^{*r})$ (see equation (3)).

   (b) Since p_{it}^d are random realizations of the price distribution, by averaging $\tilde{V}^r(s;p_{it}^d,\theta^{*r})$ across p_{it}^d's, we basically integrate out price and obtain $E_p\tilde{V}^r(s;p,\theta^{*r})$.

6. Go to iteration $r+1$.

We should note that in step 4, it may not be worthwhile to compute the pseudo-likelihood, $\rho^r(\theta^r)$, every time we reject θ^{*r}. When we reject θ^{*r}, $\theta^r = \theta^{r-1}$. So we could use $\rho^{r-1}(\theta^{r-1})$ as a proxy for $\rho^r(\theta^r)$. Our experiences suggest that we can speed up the computational time if we use this approach and compute $\rho^r(\theta^r)$ once every several successive rejections.

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 inverted gamma prior on σ_{G_j}.

Note that during the MCMC iterations, we make draws of G_{ij}^l, which is consumer i's valuation of reward at store j from the population distribution of G_{ij}. This draw is regarded as a parameter and we will use the Metropolis-Hastings algorithm to draw G_{ij}^l. As a result, conditional on G_i, the value functions do not depend on θ_2.

Each MCMC iteration mainly consists of three blocks.

1. Draw G_j and σ_{G_j} for $j = 1, 2$ (the parameters that capture the distribution of G_{ij} for the population).

2. Draw individual parameters G_{ij} for all i and $j = 1, 2$.

3. Draw the rest of the parameters, i.e., α_1, α_2, γ, and β.

The estimation steps are as follows. For each MCMC iteration (r),

1. Start with

$$H^r = \{\{\theta^{*l}, G_{i^*}^l, E_p\tilde{V}^l(.; p, G_{i^*}^l, \theta_1^l)\}_{l=r-N}^{r-1}, \rho^{r-1}(G_i^{r-1}, \theta_1^{r-1})\}$$

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_j^r (population mean of G_{ij}) from the posterior density (normal) computed by $\sigma_{G_j}^{r-1}$, and $\{G_{ij}^{r-1}\}_{i=1}^I$.

3. Draw $\sigma_{G_j}^r$ (population variance of G_i) from the posterior density (inverse gamma) computed by G_j^r and $\{G_{ij}^{r-1}\}_{i=1}^I$.

4. For each i, use the Metropolis-Hastings algorithm to draw G_{ij}^{*r}.

   (a) Draw G_{ij}^{*r} from the prior on G_{ij}, i.e., $N(G_j^r, (\sigma_{G_j}^r)^2)$.

   (b) Compute the likelihood for consumer i and determine whether or not to accept G_i^{*r}. Let G_i^r be the one that is accepted. Note that G_{ij}^{*r} will only affect $\hat{E}_p^r V(s; p, G_i^{*r}, \theta_1^l)$ for consumer i only and not for other consumers. Thus we only need to approximate the Emax functions for consumer i, using the average of $\{E_p V^l(s; p, G_{i^*}^l, \theta_1^l)\}_{l=r-N}^{r-1}$ across $G_{i^*}^l$, treating G_i^{*r} as one of the parameters when computing the weights.

   (c) Repeat for all i.

5. Draw α_1^{*r}, α_2^{*r}, γ^{*r}, and β^{*r} (candidate parameter vector).

6. 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.

(a) 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, $\{E_p V^l(s; p, G_{i^*}^l, \theta_1^l)\}_{l=r-N}^{r-1}$. In computing the weights for past value functions, we treat G_i as a parameter. Note that in the case of independent kernels, equation (5) becomes

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

In this formula, $K_h(\theta_1^{*r} - \theta_1^{r-k})$ is common across consumers. Thus, the computation time for the emax function approximation does not increase proportionally as the number of consumers increases.

(b) Compute $\tilde{V}_0^r(s; \theta_1^{*r})$, $\tilde{V}_1^r(s; p_{i1t}^d, G_i^r, \theta_1^{*r})$ and $\tilde{V}_2^r(s; p_{i2t}^d, G_i^r, \theta_1^{*r})$, by equations (5) and (6), respectively, using the approximated Emax functions computed above. These alternative specific pseudo-value functions will be used to construct the pseudo-likelihood of G_i^r and θ_1^{*r}.

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

(a) Given $\tilde{V}_0^r(s; \theta_1^{*r})$, $\tilde{V}_1^r(s; p_{i1t}^d, G_{i^*}^r, \theta_1^{*r})$ and $\tilde{V}_2^r(s; p_{i2t}^d, G_{i^*}^r, \theta_1^{*r})$, we can obtain the pseudo-value function, $\tilde{V}^r(s; p_{it}^d, G_i^{*r}, \theta_1^{*r})$.

(b) Since p_{it}^d are random realizations of the price distribution, by averaging $\tilde{V}^r$ across p_{it}^d's, we basically integrate out price and obtain $E_p \tilde{V}^r(s; p, G_{i^*}^r, \theta_1^{*r})$.

8. Go to iteration $t + 1$.

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. An

alternative way is to store pseudo-value functions individually for each consumer. That is, in each iteration we have $H^r = \{\theta^{*l}, \{G_i^l, E_p\tilde{V}^l(s; p, G_i^l, \theta^l)\forall s\}_{i=1}^I\}_{l=r-N}^{r-1}$. One advantage from storing pseudo-value functions individually is that the past pseudo-value functions used in the emax function approximation become more relevant, in the sense that they are evaluated at G_i^l's, which should be closer to the current candidate value of 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 could reduce the computation time. But one drawback is that we need much more memory to store past pseudo-value functions individually.

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 increase the acceptance rate, but might also increase the number of expected future value approximation calculations and likelihood evaluations.

3.3 Choice of bandwidth for kernel 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 increases. The following example illustrates the intuition. Note that ten points uniformly distributed in the unit interval tend to be close neighbors, but become increasingly far apart when scattered in the unit

square and cube. Thus the number of observations available to give 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.[2] 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.

In the IJC algorithm, we can mitigate this problem because, 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 function (i.e., increasing N). This is similar to the advantage of using the standard MCMC method to draw from the posterior distribution – the econometrician has control on the number of iterations that requires to obtain convergence. Thus in practice, we should expect that we need to increase N as the number of the model parameters increases. As a result, it would also take more time to compute one iteration. In fact, the nonparametric literature suggests that the convergence rate is typically inversely related to the number of dimensions. We have not characterized what the optimal relationship between N and the number of parameters should be. It is likely that the optimal relationship is model specific, as the shape of the likelihood function is also model

[2]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 increases exponentially with the number of state variables and the number of values for each state variable.

specific.

In a standard nonparametric estimation, there are a couple of ways to choose the optimal bandwidth. In computing the optimal bandwidth, we utilize the observed data. However, in the IJC algorithm, this is impossible as the set of past pseudo-value functions is not observed a priori. One way of alleviating this is as follows. We first estimate a static version of the model. Given the point estimates and the standard errors of the parameters, we generate N simulated draws of parameter vectors and use them to compute the emax values at those N simulated parameter values by contraction mapping. These N emax values serve as the data in the sense of a standard nonparametric estimation and can be used to compute the optimal bandwidth.

Another important 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).

The most important issue is that researchers need to ensure that the pseudo-value function gives us a good proxy for the true value function. We suggest that researchers check the distance between pseudo-value function and the true value function during the estimation, and adjust the bandwidth, h, and N within the iterative process. For instance, one can compute the means of the MCMC draws once every 1000 iterations, $\bar{\bar{\theta}}$, and then compare the distance between the pseudo-value function and the exact value function at $\bar{\bar{\theta}}$. If the distance is larger than what the researcher would accept, then reduce h and/or increase N, and vice versa.[3] Of course, the cost of increasing N is that it requires more memory and it would take more time to compute the weighted average. But thanks to the advance of computational power, the cost of memory is decreasing rapidly over time these days. Hence, we expect that memory would unlikely be a constraint. 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 value function at a small number of simulated heterogeneous parameter vectors, say 5. This would be equivalent to solving 5 homogeneous DDP models exactly.

4 Estimation Results

The data used in the estimation is obtained by solving and simulating the model given the predetermined set of parameter values. Throughout the estimation, the following parameters were fixed: $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\bar{p} = 1.0$, and $\sigma_p = 0.3$. We use the Gaussian

[3] If memory is not a constraint, we suggest that researchers store a large N past pseudo-value functions, 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.

kernel to weight the past pseudo-value functions in 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.

Table 1 shows the estimation results based on the homogeneous model presented above. The data is simulated based on the following parameter values: $\alpha_1 = \alpha_2 = 0.0$, $G_1 = 1.0$, $G_2 = 5.0$, $\gamma = -1.0$, and $\beta = 0.6$ or 0.8. In order to draw β by using the Random-Walk Metropolis-Hastings, we transform it as $\beta = \frac{1}{1+exp(\phi)}$ and draw ϕ instead of drawing β directly. For all parameters, flat prior is used. The posterior distributions in Table 1 indicates that the IJC algorithm was able to recover the true parameter values well for both $\beta = 0.6$ and 0.8.

Table 2 shows the estimation results based on the heterogeneous model. 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: $\alpha_1 = \alpha_2 = 0.0$, $G_1 = 1.0$, $G_2 = 5.0$, $\sigma_{G_1} = 0.0$, $\sigma_{G_2} = 1.0$, $\gamma = -1.0$, and $\beta = 0.6$ or 0.8. Again, we transform β by logit formula, i.e., $\beta = \frac{1}{1+exp(\phi)}$. For α_1, α_2, G_1, γ, and ϕ, we use flat prior. For G_2, we use a normal prior. For σ_{G_2}, we use an inverted gamma prior. The IJC algorithm again was able to recover the true parameter values well in both $\beta = 0.6$ and 0.8.

Table 3 summarizes the computation time for each of the four models estimated above. As a benchmark, we estimated the models using the full solution based Bayesian algorithm. In the full solution based Bayesian algorithm, we use 100 simulated draws of prices to integrate out the future price. Note that in the full solution based Bayesian algorithm, the computation time will increase as the discount factor becomes larger. This is because

the contraction mapping will take more steps when the discount factor is larger. 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 the 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. 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 is much more striking. 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.

As discussed above, one issue in using the IJC is how to pick up the bandwidth and N. Using the homogeneous model with $\beta = 0.8$, we investigate how changes in N influence the speed of convergence and the posterior distributions. Table 4 shows that as we increase N, the standard deviation of the posterior distribution for some parameters becomes smaller.

We have also estimated the model with $\beta = 0.98$ (results not reported here). When β approaches one, we find that it is quite difficult to separately identify α_j and G_j. The main problem is that when the discount factor is large, it does not matter much when a consumer receives the reward as we demonstrated above. As a result, G_j would simply shift the choice probabilities, similar to the way that α_j does.

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. However, it might appear that there is a limitation of this method. It is possible that when we may be 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, and one does not get a solution of the dynamic programming problem evaluated at those policy parameter vectors.

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 $\acute{\theta}_i = 100t * \theta_i$. The basic idea of this modified procedure is to generate a sequence of draws for the policy parameters and the MCMC draws of posterior distribution of actual parameters, and also solve for the value functions evaluated at simulated policy and actual parameters. Moreover, the modified procedure stores the past pseudo-value functions evaluated at both the past simulated policy and actual parameter vectors (i.e., $\acute{\theta}^{r-l}$ and θ^{r-l}).

To be more precise, the modification consists of two parts: (i) In each Metropolis-Hastings step, which we determine θ^r, we set the draw of the policy parameter vector as follows: $\acute{\theta}^r_{-i} = \theta^r_{-i}$ and $\acute{\theta}^r_i = (1+t)\theta^r_i$; (ii) use $\{E_p\tilde{V}^l(.;p,\theta^{*l})\}_{l=r-N}^{r-1}$ to form an Emax at

$\acute{\theta}^r$, and then obtain the value function and the choice probabilities at $\acute{\theta}^r$. One can then store the history of $\acute{\theta}^{r-l}$'s, and the approximated value functions and choice probabilities evaluated at $\acute{\theta}^{r-l}$'s. When the algorithm converges, this part of the outputs will represent the results of the policy experiment.

This procedure will increase the computational burden of each iteration slightly due to the calculation of the approximated emax at θ^{r+1}. However, it is relatively straightforward to implement, and requires very little extra programming effort. Moreover, before the IJC algorithm converges, there is actually no need to compute the emax at $\acute{\theta}^{r+1}$. So one can set the algorithm so that it will not start doing this extra calculation until it reaches a large number of iteration.

5.2 Continuous State Space

The state space of the reward program 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, capital stocks, etc. In this section, we describe the procedure of the IJC algorithm that extends the random grid approximation proposed by Rust (1997). For simplicity, we consider the homogeneous model here.

Consider a modified version of the reward program model without 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 (4) are conditional on current prices, $E_{p'}[V(s',p';\theta)|p]$. In the Rust random grid approxima-

tion, 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.

IJC discuss how to extend their method to combine with the Rust random grid approximation method. In the model discussed here, 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.

$$\hat{E}_{p'}^r[V(s', p'; \theta^r)|p] = \sum_{n=1}^{N} \tilde{V}^{r-n}(s', p^{r-n}; \theta^{r-n}) \frac{K_h(\theta^r - \theta^{r-n}) f(p^{r-n}|p; \theta_p)}{\sum_{k=1}^{N} K_h(\theta^r - \theta^{r-k}) f(p^{r-k}|p; \theta_p)}. \quad (6)$$

Unlike the Rust random grid approximation, the random grid points here 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 5 of section 3.2.1 needs to be modified slightly. We use a draw of p^r instead of the observed price vector, p^d, and 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,

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

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 (6).

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 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 each grid point separately.

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 state dependence 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 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.

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.

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.

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.

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.

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

		β = 0.6		β = 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), $h = 0.1$ (bandwidth).

Table 2: Estimation Results: Heterogeneous Model

		β = 0.6		β = 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), $h = 0.1$ (bandwidth).

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

	Homogeneous Model			Heterogeneous Model		
algorithm	β = 0.6	β = 0.8	β = 0.98	β = 0.6	β = 0.8	β = 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 ($\bar{S}_1 = 2$, $\bar{S}_2 = 4$).

Table 4: The Impact of N

		N=100		N=500		N=1000	
parameter	TRUE	mean	sd	mean	sd	mean	sd
α_1 (intercept for store 1)	0.0	-0.027	0.024	-0.027	0.026	-0.030	0.022
α_2 (intercept for store 2)	0.0	-0.020	0.039	-0.018	0.042	-0.018	0.028
G_1 (reward for store 1)	1.0	1.053	0.029	1.052	0.026	1.052	0.021
G_2 (reward for store 2)	5.0	5.110	0.135	5.097	0.132	5.088	0.085
γ (price coefficient)	-1.0	-1.000	0.018	-0.999	0.019	-0.996	0.019
β (discount factor)	0.8	0.801	0.008	0.800	0.010	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: $h = 0.1$ (bandwidth).

Figure 1: Choice probabilities across states for different discount factors

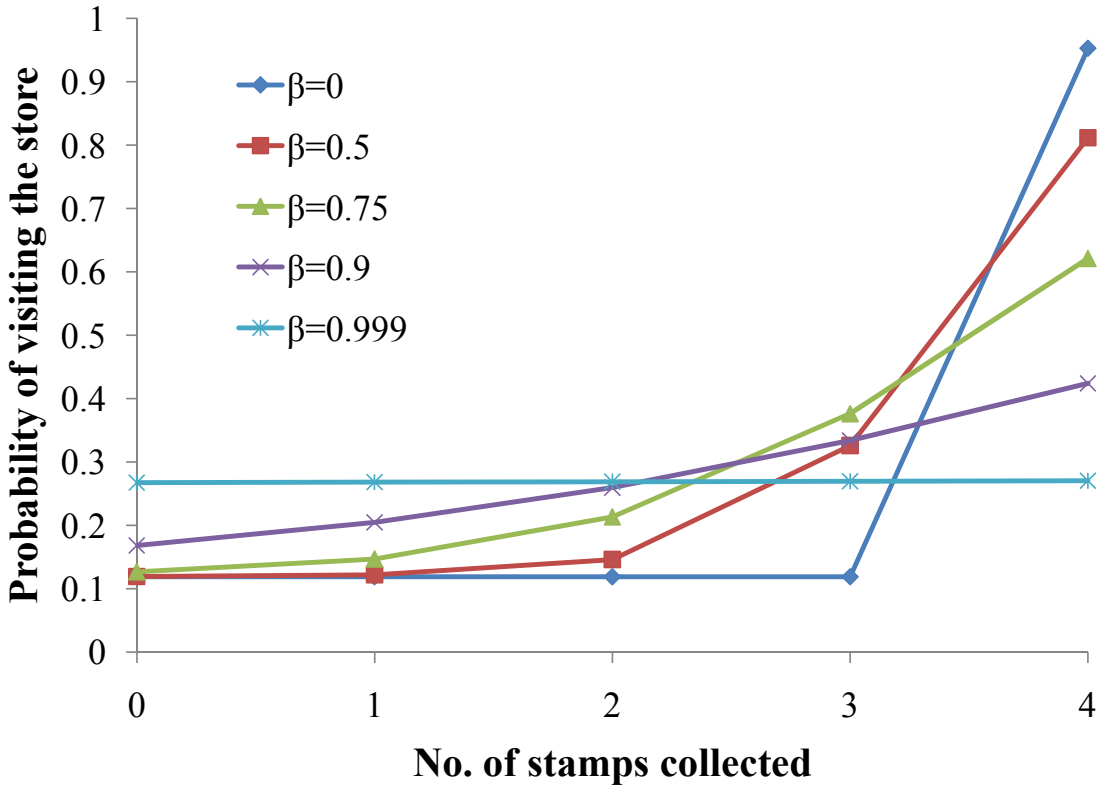
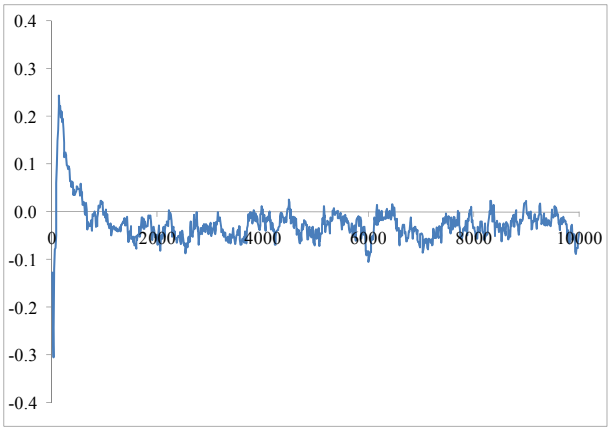
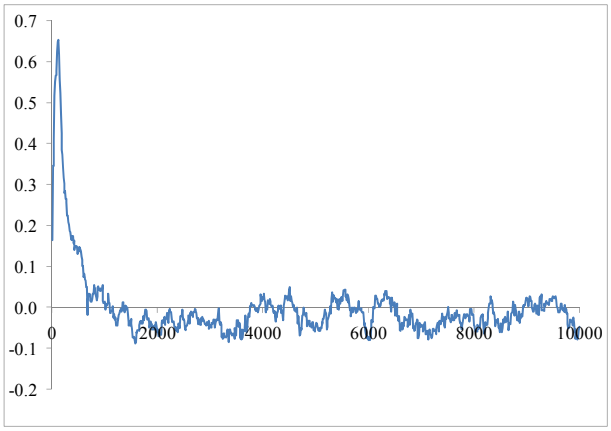


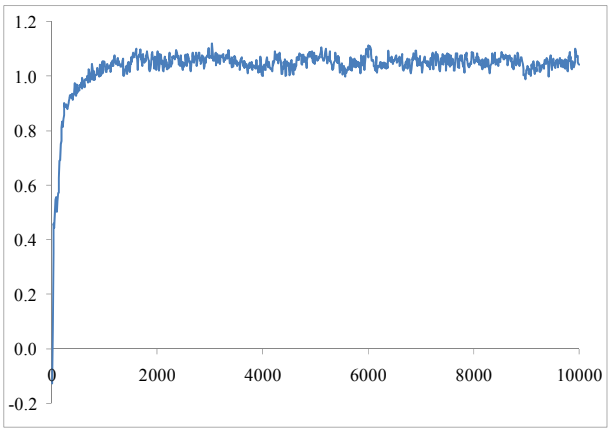
Figure 2: MCMC plots: Homogeneous Model with $\beta = 0.8$



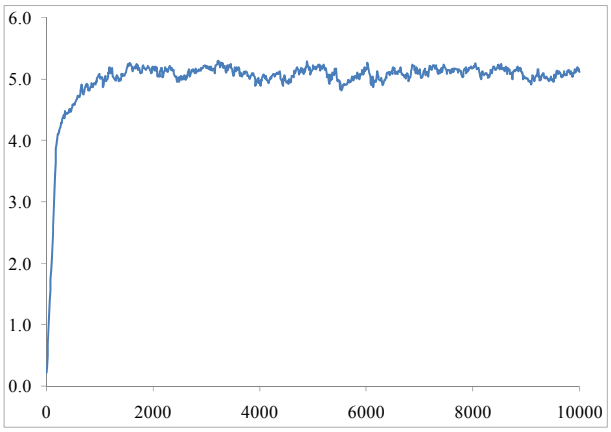
α_1 (true value = 0.0)



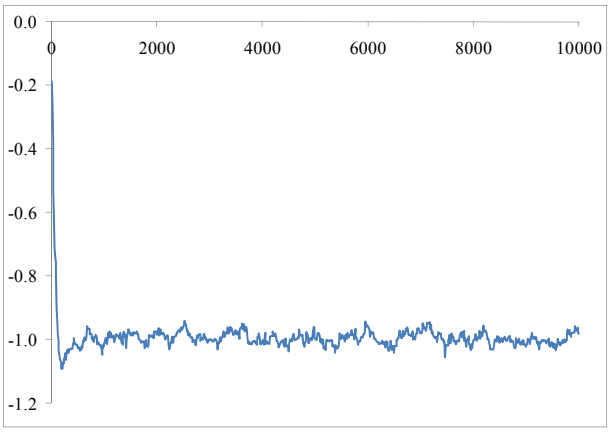
α_2 (true value = 0.0)



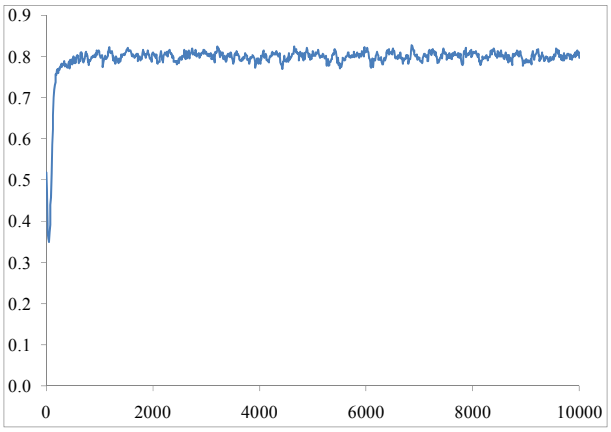
G_1 (true value = 1.0)



G_2 (true value = 5.0)

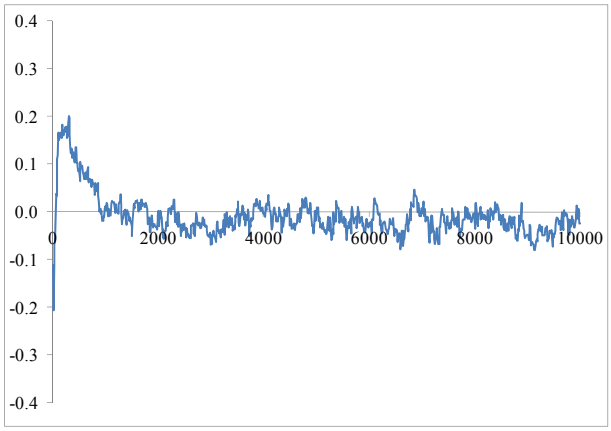


γ (true value = -1.0)

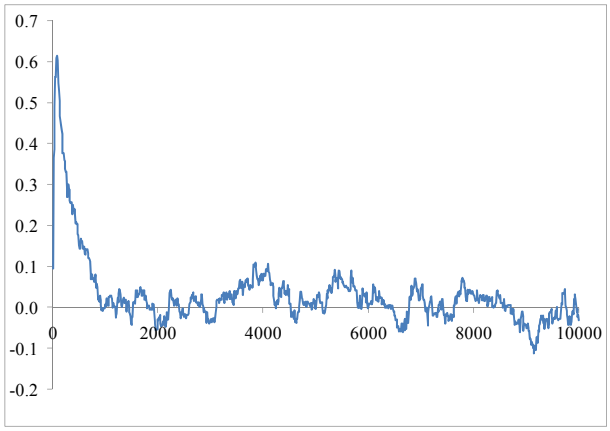


β (true value = 0.8)

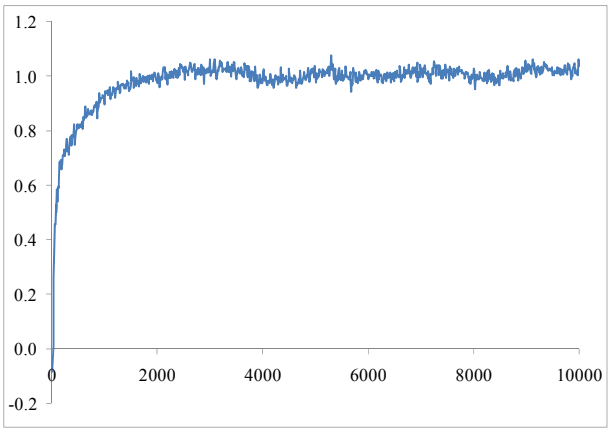
Figure 3: MCMC plots: Heterogeneous Model with $\beta = 0.8$



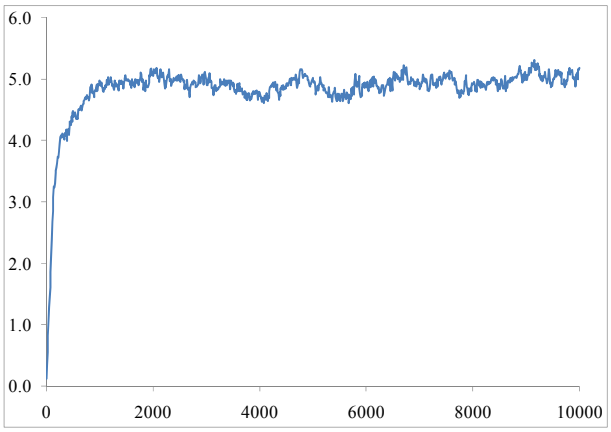
α_1 (true value = 0.0)



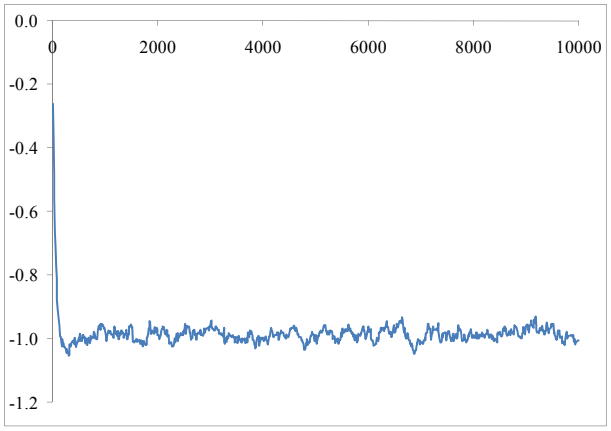
α_2 (true value = 0.0)



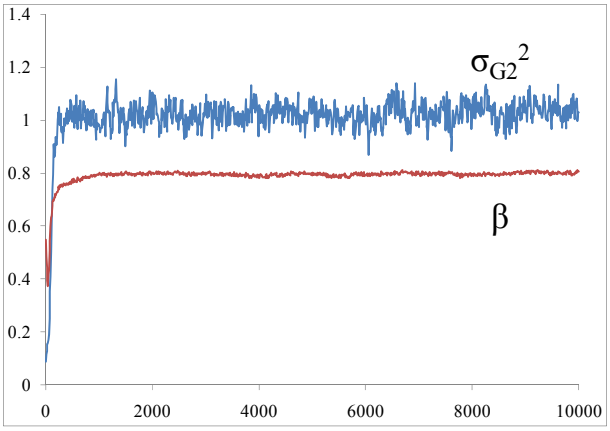
G_1 (true value = 1.0)



G_2 (true value = 5.0)



γ (true value = -1.0)



β (true value = 0.8)

σ_{G2}^2 (true value = 1.0)