Quant Mark Econ (2012) 10:151–196
DOI 10.1007/s11129-012-9119-6

A practitioner's guide to Bayesian estimation of discrete choice dynamic programming models

Andrew T. Ching · Susumu Imai · Masakazu Ishihara · Neelam Jain

Received: 29 October 2011 / Accepted: 17 January 2012 / Published online: 19 February 2012

Abstract This paper provides a step-by-step guide to estimating infinite horizon discrete choice dynamic programming (DDP) models using a new Bayesian estimation algorithm (Imai et al., Econometrica 77:1865–1899, 2009a) (IJC). 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 solve the DDP model at the current iterated parameter values. Consequently, it has the potential to significantly alleviate the computational burden of estimating DDP models. To illustrate this new estimation method, we use a simple dynamic store choice model where stores offer "frequent-buyer" type rewards programs. Our Monte Carlo results demonstrate that the IJC method is able to recover the true

The computer codes (in C and Matlab) for implementing the Monte Carlo exercises are available upon request.

A. T. Ching (✉)
Rotman School of Management, University of Toronto,
105 St George Street, Toronto, ON, Canada M5S 3E6
e-mail: aching@rotman.utoronto.ca

S. Imai
Department of Economics, Queen's University, Kingston, ON, Canada
e-mail: imais@econ.queensu.ca

M. Ishihara
Stern School of Business, New York University, New York, NY, USA
e-mail: mishihar@stern.nyu.edu

N. Jain
Department of Economics, City University London, London, UK
e-mail: neelam.jain.1@city.ac.uk

parameter values of this model quite precisely. We also show that the IJC method could reduce the estimation time significantly when estimating DDP models with unobserved heterogeneity, especially when the discount factor is close to 1.

Keywords Bayesian estimation · Dynamic programming · Discrete choice models · Rewards programs

JEL Classification C11 · C35 · C61 · D91 · M31

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 the discrete choice dynamic programming (DDP) model. This framework has been applied to study a manager's decision to replace old equipment (e.g., Rust 1987), career decision choice (e.g., Keane and Wolpin 1997; Diermeier et al. 2005), choice to commit crimes (e.g., Imai and Krishna 2004), dynamic brand choice (e.g., Erdem and Keane 1996; Gönül and Srinivasan 1996; Crawford and Shum 2005), dynamic quantity choice with stockpiling behavior (e.g., Erdem et al. 2003; Sun 2005; Hendel and Nevo 2006), new product/technology adoption decisions (e.g., Ackerberg 2003; Song and Chintagunta 2003; Yang and Ching 2010), new product introduction decisions (e.g., Hitsch 2006), dynamic pricing decisions (e.g., Ching 2010), 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 model: (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 the classical approach to estimate them. To overcome the hurdle due to the curse of dimensionality problem, several studies have proposed different ways to approximate the dynamic programming solutions (e.g., Keane and Wolpin 1994; Rust 1997; Hotz and Miller 1993; Aguirregabiria and Mira 2002; Ackerberg 2009).[1] Nevertheless, little progress has been made in handling the

[1]Geweke and Keane (2000) 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 and fully specifying the DDP model, their estimation results are not efficient and in general policy experiments cannot be conducted under their approach.

complexity of the likelihood function resulting from 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, the 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). Nonetheless, 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 the classical approach to achieve convergence. In each simulated draw of the 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 model can be solved very quickly or there exists a closed form solution (e.g., Lancaster 1997).

Recently, Imai et al. (2009a) (IJC) propose a new modified Bayesian MCMC algorithm to reduce the computational burden of estimating infinite horizon DDP models. 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 solve the DDP model at the current iterated parameter values. This new method is potentially superior to prior methods because (1) it could significantly reduce 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.

The objective of this paper is to provide a step-by-step guide for using the IJC method, and demonstrate some properties of this method. We consider an example where consumers need to choose which store to visit, and each store offers its own frequent-buyer rewards program. Our Monte Carlo results demonstrate that the IJC method is able to recover the true parameter values of this model quite precisely. We also show that the IJC method could reduce the estimation time very significantly when estimating DDP models with unobserved heterogeneity, especially when the discount factor is close to 1.

The rest of the paper is organized as follows. In Section 2, we present the IJC method and a general class of models that it can handle. In Section 3, we describe the simple dynamic store choice model with rewards programs. In Section 4, we discuss how to use the IJC method to estimate this model in detail. We also discuss several practical aspects of using this new method. Section 5 conducts two sets of Monte Carlo experiments to demonstrate the performance and some properties of the IJC algorithm. Section 6 compares the IJC algorithm with other approximation approaches to estimate DDP models. Section 7 is the conclusion.

2 General description of the IJC algorithm

2.1 The basic framework for a stationary DDP problem

We consider an infinite horizon stationary dynamic model of a forward-looking agent. By "stationary," we mean that conditioning on the value of the state variables, the optimal decisions of the agent do not depend on time t. Let $\mathcal{S}$ be the set of state space points and let s be an element of $\mathcal{S}$. We assume that $\mathcal{S}$ is finite. Let A be the finite set of all possible actions and let a be an element of A.

Let Θ be the parameter space and let $\theta = (\theta_R, \theta_\epsilon, \theta_s, \beta) \in \Theta$ be the parameter vector. Let $R(s, a, \epsilon_a; \theta_R)$ be the current period return function of choosing action a given s, and ϵ_a is a random shock to current returns to choice a. Depending on the applications, the return could be consumers' utility, wages, revenue, etc. Let ϵ be a vector whose ath element is ϵ_a. We assume that ϵ follows a multivariate distribution $F(\epsilon; \theta_\epsilon)$ with density function $dF(\epsilon; \theta_\epsilon)$ and is independent over time. But note that ϵ could be correlated across a's. We assume that the transition probability of next period state s', given current period state s and action a, is $f(s'|s, a; \theta_s)$. Let β be the discount factor. We define the time invariant value function to be the maximum of the discounted sum of expected returns:

$$\mathcal{V}(s_t, \epsilon_t; \theta) \equiv \max_{\{a_t, a_{t+1}, \ldots\}} E\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} R(s_\tau, a_\tau, \epsilon_{a_\tau}; \theta_R) | s_t, \epsilon_t\right].$$

If we further assume that $0 < \beta < 1$ and R is bounded, then the time invariant value function will satisfy the following functional equation.

$$\mathcal{V}(s, \epsilon; \theta) = \max_{a \in A}\{R(s, a, \epsilon_a; \theta_R) + \beta E_{s', \epsilon'}[\mathcal{V}(s', \epsilon'; \theta)|s, a]\}, \tag{1}$$

where the expectation is taken over the next period s' and ϵ':

$$E_{s', \epsilon'}[\mathcal{V}(s', \epsilon'; \theta)|s, a] = \sum_{s' \in S}\left[\int_{\epsilon'} \mathcal{V}(s', \epsilon'; \theta) dF(\epsilon'; \theta_\epsilon)\right] f(s'|s, a; \theta_s). \tag{2}$$

The literature refers to Eq. 1 as the Bellman equation, and $E_{s', \epsilon'}[\mathcal{V}(s', \epsilon'; \theta)|s, a]$ as the expected value function. If we use the Bellman equation to define a functional operator (which maps a function to another function), we can rewrite Eq. 1 as the fixed point of the Bellman operator, i.e.,

$$\mathcal{V}(s,\epsilon;\theta) = T_\theta \mathcal{V}(s,\epsilon;\theta),$$

where

$$T_\theta \mathcal{V}(s,\epsilon;\theta) \equiv \max_{a\in A}\{R(s,a,\epsilon_a;\theta_R) + \beta E_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta)|s,a]\}.$$

It can be shown that this operator is a contraction mapping. Therefore, to solve for $\mathcal{V}$, one can start with any bounded function, $\mathcal{V}^0$, and recursively apply the Bellman operator, $\mathcal{V}^{n+1} = T_\theta \mathcal{V}^n$. In the limit, $\mathcal{V}^n \to \mathcal{V}$. This procedure, which is called *the method of successive approximation*, provides a tractable way to solve for the Bellman equation and agents' optimal decisions numerically.

Suppose we only observe agents' choice and their corresponding state, $\{\mathsf{a}_i, \mathsf{s}_i\}_{i=1}^I$, where i indexes agents (I agents), $\mathsf{a}_i = \{\mathsf{a}_{it}\}_{t=1}^T$, and $\mathsf{s}_i = \{\mathsf{s}_{it}\}_{t=1}^T$. We assume that ϵ_{iat} are unobserved to researchers. To estimate θ, we need to construct the likelihood of the observed data. The likelihood will take the form of products of choice probabilities, $P_j(\mathsf{s}_{it};\theta)$. Let $V_j(s,\epsilon_j;\theta) \equiv R(s,j,\epsilon_j;\theta_R) + \beta E_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta)|s,a=j]$. V_j is usually called the *alternative specific value function*. The probability of observing agent i choosing j given the observed state, s_{it}, is:

$$P_j(\mathsf{s}_{it};\theta) = P\left(\epsilon : V_j(\mathsf{s}_{it},\epsilon_{ijt};\theta) \geq V_l(\mathsf{s}_{it},\epsilon_{ilt};\theta), \forall l\right).$$

Then the likelihood increment for agent i at time t is:

$$L_{it}(\mathsf{a}_{it}|\mathsf{s}_{it};\theta) = P_j(\mathsf{s}_{it};\theta) \text{ if } \mathsf{a}_{it} = j.$$

The likelihood of the observed data is

$$L(\mathsf{a}|\mathsf{s};\theta) = \prod_i \prod_t L_{it}(\mathsf{a}_{it}|\mathsf{s}_{it};\theta),$$

where $\mathsf{a} = (\mathsf{a}_1,\cdots,\mathsf{a}_I)$, $\mathsf{s} = (\mathsf{s}_1,\cdots,\mathsf{s}_I)$. Let $\pi(\theta)$ be the prior distribution. Then the posterior distribution of the parameters is proportional to the product of prior and the likelihood. That is,

$$p(\theta|\mathsf{a},\mathsf{s}) \propto \pi(\theta)\cdot L(\mathsf{a}|\mathsf{s};\theta).$$

2.2 The IJC algorithm

To simplify the exposition, we assume that θ_s is known.[2] A structural estimation algorithm can typically be broken down into two components: outer loop and inner loop. In Bayesian estimation, the outer loop is implemented using the Markov Chain Monte Carlo (MCMC) method (e.g., Metropolis-Hastings

[2]Given that we assume s evolves according to $f(s'|s,a;\theta_s)$, one can estimate θ_s based on the observed transition of s alone, without using the DDP model.

Algorithm).[3] It makes use of $\pi(\theta)$ and $L(\mathsf{a}|\mathsf{s};\theta)$ to obtain a new draw from the posterior distribution. The inner loop, which is nested within the outer loop, computes $\mathcal{V}(.,\epsilon;\theta)$, and $V_j(\mathsf{s}_{it},\epsilon_j;\theta)$ for constructing the likelihood. Given a trial parameter vector θ^*, the traditional approach applies the Bellman operator repeatedly to solve for $\mathcal{V}(.,\epsilon;\theta^*)$, which is the unique fixed point of the contraction mapping. Since Rust (1987), the literature refers to this approach as the Nested Fixed Point Algorithm. An MCMC outer loop typically requires many iterations to achieve convergence (10,000–100,000 iterations). This (roughly) implies that we need to solve for the fixed point 10,000–100,000 times if we apply the Nested Fixed Point Algorithm to conduct Bayesian inference. Obviously, the traditional approach is computationally very demanding.

The IJC algorithm aims at providing a more practical way for Bayesian estimation of DDP models. It relies on the following insights. In most of the applications, the value function is continuous in θ.[4] Therefore, at any given parameter vector, θ^*, it is possible to approximate the expected value function $(E_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta^*)|s,a])$ using a set of value functions obtained from the earlier iterations of the MCMC algorithm. Note that this set of value functions are evaluated at randomly drawn θ's and ϵ's. Intuitively, one can obtain an estimate of the expected value function given any current trial θ by running a non-parametric regression.

Another insight is the following. For any given neighborhood of the parameter space that belongs to the support of the proposal density, the outer loop (i.e., the MCMC algorithm) will revisit this neighborhood infinitely often as the number of outer loop iterations goes to infinity. Therefore, in each iteration, one does not have to apply the Bellman operator iteratively until we solve for the value function at a given trial parameter vector. At the minimum, one needs to apply the Bellman operator only once in each iteration, and then save this "*pseudo*-value function" for future use when we revisit this neighborhood. As the algorithm runs, we will revisit this neighborhood repeatedly, and this implies that the "effective" number of Bellman operator iterations applied to this neighborhood will increase. Since we can make the neighborhood arbitrarily small, it follows from the contraction mapping property of the Bellman operator that the *pseudo*-value functions converge to the true value functions for any θ that belongs to the support of the proposal density. When this convergence is achieved, this algorithm effectively "converges" to the conventional Bayesian MCMC algorithm. By just applying the Bellman operator once in each iteration, this approach could significantly reduce the computational time per iteration.[5]

[3]Walsh (2004) provides an excellent introduction to MCMC methods.

[4]Norets (2010) provides a set of general model assumptions under which the implied value function is continuous in θ.

[5]Formally, the convergence results require three more assumptions: (i) Θ is compact; (ii) the return function, $R(s,a,\epsilon;\theta_R)$ and the initial guess of the value function, $\mathcal{V}^0(s,\epsilon;\theta)$, are continuous in ϵ and θ; (iii) the prior distribution $\pi(\theta)$ is positive and bounded for any given $\theta\in\Theta$.

We will now outline the basic IJC algorithm. Since the IJC algorithm approximates the value function, we will use the terms, pseudo-value function ($\tilde{\mathcal{V}}^l$), pseudo-expected value function ($\tilde{E}^l_{s',\epsilon'}\mathcal{V}$), pseudo-alternative specific value function ($\tilde{V}^l_j$) and pseudo-likelihood ($\tilde{L}^l$) when we outline the algorithm. The superscript l indicates that they are derived at outer loop iteration l.

(1) The outer loop (Metropolis-Hastings (M-H) Algorithm)

The outer loop is a Metropolis-Hastings (M-H) algorithm, which is a version of Markov-chain Monte Carlo (MCMC) algorithm. The M-H algorithm allows us to simulate a Markov-chain of parameter draws $\{\theta^l\}$ whose sample distribution converges to the true posterior distribution. It is well-known that directly drawing θ^l from the posterior density $p(\theta|\mathsf{a}, \mathsf{s})$ is computationally difficult due to the complexity of the high dimensional posterior density. MCMC provides a tractable way to by-pass this problem. In the M-H algorithm, we first draw a candidate parameter vector from a proposal density. Then, we decide whether or not to accept the candidate parameter vector. We will denote the candidate parameter vector in iteration r by θ^{*r} and the accepted parameter vector in iteration r by θ^r. Let $q(\theta^{*r}|\theta^{r-1})$ denote the proposal density of θ^{*r} given θ^{r-1} (e.g., $\theta^{*r} \sim N(\theta^{r-1}, \sigma^2)$).

Suppose we are at the beginning of iteration r. We first draw a candidate parameter vector from the proposal density

$$\theta^{*r} \sim q\left(\theta^{*r}|\theta^{r-1}\right).$$

Then, accept θ^{*r} with probability λ. That is,

$$\theta^r = \begin{cases} \theta^{*r} & \text{with probability } \lambda \\ \theta^{r-1} & \text{with probability } 1-\lambda \end{cases}$$

where λ, the acceptance probability, is calculated as follows.

$$\lambda = \min\left\{ \frac{\pi(\theta^{*r})\tilde{L}^r(\mathsf{a}|\mathsf{s};\theta^{*r})q(\theta^{r-1}|\theta^{*r})}{\pi(\theta^{r-1})\tilde{L}^r(\mathsf{a}|\mathsf{s};\theta^{r-1})q(\theta^{*r}|\theta^{r-1})}, 1 \right\},$$

where $\tilde{L}$ is the pseudo-likelihood function based on a pseudo-alternative specific value function, which is computed in the inner loop.

(2) The inner loop

The inner loop does two jobs. First, it computes the pseudo-alternative specific value function, which is then used to construct the pseudo-likelihood in the outer loop. Second, it updates the pseudo-value function by applying the Bellman operator once. In these processes, we need to first compute the expected value function. To alleviate the computational burden of deriving the fixed point of the above dynamic programming algorithm, IJC approximate the expected value function by using the information from earlier iterations of the MCMC estimation algorithm. They propose to store $H^r = \{\theta^{*l}, \tilde{\mathcal{V}}^l(., \epsilon^l; \theta^{*l})\}_{l=r-N}^{r-1}$, where ϵ^l is a simulated draw obtained from $F(\epsilon; \theta^{*l}_\epsilon)$

in iteration l; N is the number of past value functions to store. The pseudo-expected value function is then

$$\tilde{E}^r_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta^{*r})|s,a] = \sum_{s'\in S}\left[\sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}^l(s',\epsilon^l;\theta^{*l})\cdot\omega(\theta^{*l},\theta^{*r})\right] f(s'|s,a;\theta_s), \tag{3}$$

where $\omega(\theta^{*l},\theta^{*r})$ is the weight that takes high (low) value for θ^{*l} that is close to (far away from) the current θ^{*r}. In particular, IJC use the following weight,

$$\omega(\theta^{*l},\theta^{*r}) = \frac{K_h(\theta^{*l},\theta^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta^{*k},\theta^{*r})},$$

where K_h is the Gaussian kernel density with bandwidth h. We should emphasize that the step above allows us to integrate out ϵ' as well. To see this more clearly, let us fix the parameters at each iteration to θ^*, then Eq. 3 becomes

$$\begin{aligned}\tilde{E}^r_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta^{*})|s,a] &= \sum_{s'\in S}\left[\sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}^l(s',\epsilon^l;\theta^{*})\cdot\omega(\theta^{*},\theta^{*})\right] f(s'|s,a;\theta_s)\\ &= \sum_{s'\in S}\left[\frac{1}{N}\sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}^l(s',\epsilon^l;\theta^{*})\right] f(s'|s,a;\theta_s),\end{aligned}$$

where $\frac{1}{N}\sum_{l=r-N}^{r-1}\tilde{\mathcal{V}}^l(s',\epsilon^l;\theta^*)$ is the Monte Carlo approximation of the expectation with respect to ϵ'. With this pseudo-expected value function, we can construct the value of choosing each alternative conditional on (s,ϵ) (i.e., pseudo-alternative specific value function):

$$\tilde{V}^r_j(s,\epsilon_j;\theta^{*r}) = R(s,j,\epsilon_j;\theta^{*r}_R) + \beta\tilde{E}^r_{s',\epsilon'}[\mathcal{V}(s',\epsilon';\theta^{*r})|s,a=j],$$

which can then be used to construct the pseudo-likelihood for θ^{*r}. The pseudo-likelihood for θ^{r-1} can be constructed in a similar way.

To update the pseudo-value function, we simulate a draw of ϵ^r from $F(\epsilon;\theta^{*r}_\epsilon)$, substitute it into $\tilde{V}^r_j$ above for all $s\in S$, and obtain

$$\tilde{\mathcal{V}}^r(s,\epsilon^r;\theta^{*r}) = \max_{a\in A}\tilde{V}^r_a(s,\epsilon^r_a;\theta^{*r}),\quad \forall s\in S.$$

We then store $\{\theta^{*r},\tilde{\mathcal{V}}^r(.,\epsilon^r;\theta^{*r})\}$ and update H^r to H^{r+1}.

IJC prove that under some regularity conditions, as the total number of outer loop iterations grows, pseudo-value functions converge to the true value functions, and consequently the sample distribution of the accepted draws of parameter vectors converges to the true posterior distribution.[6] In theory, we need to increase N as the number of iterations grows to obtain the

[6]Strictly speaking, parameter vector draws obtained from the IJC algorithm are not a Markov chain because the pseudo-expected value function depends on the past pseudo-value functions, which are evaluated at $\{\theta^l\}_{l=r-N}^{r-2}$ in addition to θ^{r-1}. As a result, the proof of convergence is non-standard (Imai et al. 2009b).

convergence. When we have more “data points” for the non-parametric approximation, we also need the bandwidth to shrink in order to get convergence. IJC derive the precise convergence rates.[7]

We should emphasize that the pseudo-expected value function step (i.e., Eq. 3) is the key innovation of IJC. In principle, this step is also applicable in classical estimation methods such as GMM and maximum likelihood.[8] However, there are at least two advantages of implementing IJC’s pseudo-expected value function approach in Bayesian estimation.[9] First, the non-parametric approximation would be more efficient if the past pseudo-value functions are evaluated at θ^{*l}’s $(l < r)$, which 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 the parameter space along a particular path, not giving much variation in the parameters around the approximation point, resulting in poor nonparametric approximation. We therefore believe that the approximation step proposed by IJC should perform better under the Bayesian MCMC approach.[10] 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 turn out to be better estimators of the true parameter values than classical point estimates. This is because in practice, accurately simulating a posterior is usually easier than finding the global maximum/minimum of a complex likelihood/GMM objective function (e.g., Geweke et al. 2001).

We make three remarks here. First, the above approximation of the expected value function relies on a “moving window” of past pseudo-value functions, and in particular, we need to discard the “old” ones and use the most recent ones. This is because the pseudo-value function produced in each new iteration represents an improved approximation of the true value function. By discarding the “old” pseudo-value functions, it ensures that we use the most accurate past pseudo-value functions to form a weighted average

[7]Norets (2009) derives the convergence rates under the nearest neighbor kernel.

[8]Brown and Flinn (2011) 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.

[9]It is important to note that Bernstein and von Mises Theorem states that Bayesian posterior mean and the ML estimators are asymptotically equivalent.

[10]A stochastic optimization algorithm, simulated annealing, has recently gained some attention to handle complicated objective functions. This algorithm is an adaptation of the M-H algorithm (Černý 1985; Kirkpatrick et al. 1983). The approximation step proposed by IJC should also be well-suited when researchers use simulated annealing to maximize/minmize the objective function in classical approaches (e.g., ML and GMM). 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.

approximation for the expected value function. Second, note that we store the past pseudo-value functions evaluated at θ^{*l} (the draw from the proposal distribution) instead of θ^l (the draw from the posterior distribution of θ). Why do we take this approach? If we store $\{\theta^l, \tilde{\mathcal{V}}^l(., \epsilon^l; \theta^l)\}_{l=r-N}^{r-1}$, there may be a 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 value function, it is often more efficient to have a set of $\tilde{\mathcal{V}}^l$'s evaluated at parameter vectors that span the parameter space. Since θ^{*l}'s are drawn from a candidate generating function, it is much easier for us to achieve this goal by storing $\{\theta^{*l}, \tilde{\mathcal{V}}^l(., \epsilon^l; \theta^{*l})\}_{l=r-N}^{r-1}$.[11] Third, we emphasize that $\tilde{\mathcal{V}}^l(., \epsilon^l; \theta^{*l})$ means "$\tilde{\mathcal{V}}^l(s, \epsilon^l; \theta^{*l}), \forall s$". Therefore, if the size of the state space is large, this algorithm will need a large amount of computer memory to implement efficiently.

2.3 Special case: ϵ is *i.i.d.* extreme value distributed

So far, the assumptions we made for the distribution of ϵ are fairly general. In many applications, however, researchers are willing to assume that ϵ's are *i.i.d.* extreme value distributed and enter the return functions in an additively separable way. Hence,

$$R(s, a, \epsilon_a; \theta_R) = \bar{R}(s, a; \theta_R) + \epsilon_a.$$

Sometimes even though such an assumption is not strongly motivated by the modeling needs, it may still be worth adding them because it can lead to a smooth and closed form likelihood function given the value function. Another advantage is that the expected maximum of alternative specific value functions with respect to ϵ_a has a closed form expression. This will significantly reduce the computational burden of solving for the value function.[12]

We now modify our exposition about the recursive set up of the DDP model to show how to take advantage of this distributional assumption. Recall that the alternative specific value function is:

$$\begin{aligned} V_j(s, \epsilon_j; \theta) &= R(s, j, \epsilon_j; \theta_R) + \beta E_{s', \epsilon'}[\mathcal{V}(s', \epsilon'; \theta)|s, a = j] \\ &= \bar{R}(s, j; \theta_R) + \epsilon_j + \beta E_{s'}\left[E_{\epsilon'}[\mathcal{V}(s', \epsilon'; \theta)]|s, a = j\right]. \end{aligned} \tag{4}$$

[11] Imai et al. (2009b) only proved convergence for the algorithm where the value functions for the candidate parameter draws were stored. This is because it is easier to prove convergence when the stochastic variations of the parameters are controlled by the candidate generating function than jointly by the candidate generating function and the acceptance rate of the M-H algorithm.

[12] Note that in this setup, the return function is unbounded because ϵ_a has unbounded support. Therefore, to show that the Bellman operator is a contraction mapping, one needs to apply a generalized version of Blackwell's Theorem provided in Rust (1988).

Let $W_j(s;\theta) \equiv V_j(s,\epsilon_j;\theta) - \epsilon_j$. The assumption that ϵ_j's are *i.i.d.* extreme value distributed implies

$$\begin{aligned} E_{\epsilon'}[\mathcal{V}(s',\epsilon';\theta)] &= E_{\epsilon'}[\max_j V_j(s',\epsilon'_j;\theta)] \\ &= E_{\epsilon'}[\max_j\{W_j(s';\theta)+\epsilon'_j\}] \\ &= \ln\left[\sum_j \exp(W_j(s';\theta))\right]. \end{aligned}$$

We refer to $E_{\epsilon'}[\mathcal{V}(s',\epsilon';\theta)]$ as the $E_\epsilon max$ function. To simplify notation, we define $\mathcal{W}(s;\theta) \equiv E_\epsilon[\mathcal{V}(s,\epsilon;\theta)]$. Then, the solution to this stationary dynamic optimization problem can be reformulated as follows.

$$\mathcal{W}(s;\theta) = \ln\left[\sum_j \exp(W_j(s;\theta))\right], \tag{5}$$

where

$$W_j(s;\theta) = \bar{R}(s,j;\theta_R) + \beta E_{s'}[\mathcal{W}(s';\theta)|s,a=j]. \tag{6}$$

The above formulation suggests that we can store $\mathcal{W}$ instead of $\mathcal{V}$ when implementing the IJC algorithm. More precisely, we store $H^r = \{\theta^{*l}, \tilde{\mathcal{W}}^l(.;\theta^{*l})\}_{l=r-N}^{r-1}$, where $\tilde{\mathcal{W}}^l(.;\theta)$ is the pseudo-$E_\epsilon max$ function. Then the pseudo-expected value function at iteration r given the candidate parameter vector θ^{*r} will be given by,

$$\tilde{E}^r_{s'}[\mathcal{W}(s';\theta^{*r})|s,a] = \sum_{s'\in S}\left[\sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s';\theta^{*l})\cdot\omega(\theta^{*l},\theta^{*r})\right] f(s'|s,a;\theta_s).$$

With this pseudo-expected value function, we can construct the counterpart of W^r_j as follows.

$$\tilde{W}^r_j(s;\theta^{*r}) = \bar{R}(s,j;\theta^{*r}_R) + \beta\tilde{E}^r_{s'}[\mathcal{W}(s';\theta^{*r})|s,a=j],$$

and the pseudo-likelihood increment for agent i at time t is

$$\tilde{L}^r_{it}(\mathsf{a}_{it}|\mathsf{s}_{it};\theta^{*r}) = \frac{\exp(\tilde{W}^r_j(\mathsf{s}_{it};\theta^{*r}))}{\sum_{k\in A}\exp(\tilde{W}^r_k(\mathsf{s}_{it};\theta^{*r}))} \text{ if } \mathsf{a}_{it}=j.$$

To update the pseudo-$E_\epsilon max$ function, we no longer need to simulate a draw of ϵ^r. We can simply compute

$$\tilde{\mathcal{W}}^r(s;\theta^{*r}) = \ln\left[\sum_j \exp(\tilde{W}^r_j(s;\theta^{*r}))\right], \quad \forall s\in S.$$

One can then update H^r to H^{r+1}.

2.4 Extension 1: continuous state variables

So far, we have maintained the assumption that $\mathcal{S}$ is finite. However, in many marketing and economics applications, we have to deal with continuous state variables such as prices, advertising expenditures, capital stocks, etc. IJC also describe how to extend the algorithm to allow for continuous state variables, by combining it with the random grid approximation proposed by Rust (1997). To illustrate the procedure, we now assume that all state variables are continuous.[13] Let $f(s'|s, a; \theta_s)$ be the transition density of next period state s', given current period state s and action a.

To incorporate continuous state variables, we store $H^r = \{\theta^{*l}, s^l, \tilde{\mathcal{V}}^l(s^l, \epsilon^l; \theta^{*l})\}_{l=r-N}^{r-1}$, where ϵ^l is a simulated draw obtained from $F(\epsilon; \theta_\epsilon)$ in iteration l, and s^l is a simulated draw from a uniform distribution that covers the support of $\mathcal{S}$. IJC propose to construct the pseudo-expected value function in iteration r given θ^{*r} as,

$$\tilde{E}^r_{s',\epsilon'}[\mathcal{V}(s', \epsilon'; \theta^{*r})|s, a] = \sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}^l(s^l, \epsilon^l; \theta^{*l}) \cdot \psi(\theta^{*l}, \theta^{*r}; s^l, s|a)$$

where

$$\psi(\theta^{*l}, \theta^{*r}; s^l, s|a) = \frac{K_h(\theta^{*l}, \theta^{*r}) f(s^l|s, a; \theta_s)}{\sum_{k=r-N}^{r-1} K_h(\theta^{*k}, \theta^{*r}) f(s^k|s, a; \theta_s)}.$$

Thus, $\psi(\theta^{*l}, \theta^{*r}; s^l, s|a)$ assigns higher weights not only for parameters θ^{*l}'s that are close to the current parameter θ^{*r}, but also for states s^l's that have higher transition densities from state s. Since s^l's are drawn from a uniform distribution, this weighted average will integrate out the next period state s'.

With this pseudo-expected value function, we can construct the pseudo-alternative specific value functions:

$$\tilde{V}^r_j(s, \epsilon_j; \theta^{*r}) = R(s, j, \epsilon_j; \theta^{*r}_R) + \beta \tilde{E}^r_{s',\epsilon'}[\mathcal{V}(s', \epsilon'; \theta^{*r})|s, a = j],$$

which can then be used to construct the pseudo-likelihood.

To update the pseudo-value function, we simulate a draw of ϵ^r from $F(\epsilon; \theta^{*r}_\epsilon)$ and a draw of s^r from a uniform distribution over the support of $\mathcal{S}$, substitute them into $\tilde{V}^r_j$, and obtain

$$\tilde{\mathcal{V}}^r(s^r, \epsilon^r; \theta^{*r}) = \max_{a \in A} \tilde{V}^r_a(s^r, \epsilon^r_a; \theta^{*r}).$$

One important difference from the base case is that we only need to update and store the pseudo-value function at $s = s^r$, in contrast to updating and storing the pseudo-value function at all $s \in \mathcal{S}$.

[13]In general, the state space can consist of a mixture of discrete and continuous state variables. In such a case, readers can combine the results in the base case and in this subsection to obtain the nonparametric approximation of the expected value function. See Section 4.4 for an example.

Conventionally, randomly generated grid points are fixed throughout the solution-estimation algorithm. IJC, however, propose to randomly draw one state vector, s^r, in iteration r, apply the Bellman operator once and store the pseudo-value function conditional on s^r. Thus, when approximating the expected value function, we use past pseudo-value functions that are evaluated at randomly generated s^l's. In this approach, one can easily adjust the precision of the approximation because the total number of random grid points can be made arbitrarily large by increasing N. We should also point out that if researchers simply apply the conventional Rust's random grid approximation with M fixed grid points in the IJC algorithm, they need to compute the pseudo-value functions at M grid points in each iteration. As a result, the "effective" size of the state space will become M, while the IJC's random grid approach will keep it at one. The main advantage of the IJC's random grid algorithm comes from the fact that the integration of the continuous state variables is already incorporated into the computation of the weighted average of past pseudo-value functions. This feature allows us to compute the pseudo-value function at only one grid point per iteration.

Finally, the IJC algorithm can also handle the situation in which the transition of a state variable is deterministic. A critical assumption when applying the random grid approximation by Rust (1997) is that the transition density $f(s'|s,a;\theta_s)$ is not degenerate. IJC propose a kernel-based local interpolation approach to handle this issue. Let K_{h_s} be the kernel function with bandwidth h_s for the state variable s and K_{h_θ} be the kernel function with bandwidth h_θ for the parameter vector θ. The pseudo-expected value function in iteration r given θ^{*r} can now be obtained by

$$\tilde{E}^r_{\epsilon'}[\mathcal{V}(s',\epsilon';\theta^{*r})|s,a] = \sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}^l(s^l,\epsilon^l;\theta^{*l}) \cdot \varphi(\theta^{*l},\theta^{*r};s^l,s'),$$

where

$$\varphi(\theta^{*l},\theta^{*r};s^l,s') = \frac{K_{h_\theta}(\theta^{*l},\theta^{*r})K_{h_s}(s^l,s')}{\sum_{k=r-N}^{r-1} K_{h_\theta}(\theta^{*k},\theta^{*r})K_{h_s}(s^k,s')}.$$

With this pseudo-expected value function, we can construct the value of choosing each alternative conditional on (s,ϵ):

$$\tilde{V}^r_j(s,\epsilon_j;\theta^{*r}) = R(s,j,\epsilon_j;\theta^{*r}_R) + \beta \tilde{E}^r_{\epsilon'}[\mathcal{V}(s',\epsilon';\theta^{*r})|s,a=j],$$

which can then be used to construct the pseudo-likelihood.

We update the pseudo-value function in a way similar to the case of stochastic continuous state variables. We simulate a draw of ϵ^r from $F(\epsilon;\theta^{*r}_\epsilon)$ and s^r from a uniform distribution, substitute them into $\tilde{V}^r_j$, and obtain

$$\tilde{\mathcal{V}}^r(s^r,\epsilon^r;\theta^{*r}) = \max_{a\in A} \tilde{V}^r_a(s^r,\epsilon^r_a;\theta^{*r}).$$

2.5 Extension 2: unobserved heterogeneity

Extension of the above estimation methodology to a model with individual-specific unobserved heterogeneity is straightforward. IJC propose to combine it with the Hierarchical Bayes approach to estimate individual-specific unobserved heterogeneity (e.g., Allenby and Rossi 2006). If readers are not familiar with the Hierarchical Bayes approach, we suggest them to read Chapter 12 of Train (2003) or Chapter 5 of Rossi et al. (2005) first. Although we try to make the presentation self-contained, it will be easier to follow if readers have such knowledge.

Suppose that some of the parameters for the return function, θ_R, take different values for different agents. We rewrite the parameter vector for the return function as $\theta_{Ri} = (\theta_{R1i}, \theta_{R2})$ where θ_{R1i} is a vector of individual-specific parameters and θ_{R2} is a vector of parameters common across agents. We assume that θ_{R1i} follows the density function below:

$$\theta_{R1i} \sim g(\theta_{R1i}; \mu),$$

where μ is a parameter vector for this density (it is called hyperparameter in the Bayesian inference literature). The entire parameter vector consists of $\theta = (\mu, \{\theta_{R1i}\}_{i=1}^{I}, \theta_{R2}, \theta_\epsilon, \theta_s)$. Let us rewrite this vector as $\theta = (\mu, \{\theta_{R1i}\}_{i=1}^{I}, \theta_c)$ where $\theta_c = (\theta_{R2}, \theta_\epsilon, \theta_s)$ is the vector of parameters common across agents. Then the Bellman equation in the base case (Eq. 1) can be modified as:

$$\mathcal{V}(s, \epsilon; \theta_{R1i}, \theta_c) = \max_{a \in A}\{R(s, a, \epsilon_a; \theta_{Ri}) + \beta E_{s', \epsilon'}[\mathcal{V}(s', \epsilon'; \theta_{R1i}, \theta_c)|s, a]\}.$$

Note that the value function, $\mathcal{V}(s, \epsilon; \theta_{R1i}, \theta_c)$, is now individual-specific and does not depend on θ_{R1k} for $k \neq i$. Also, it does not depend on the hyperparameter μ. Consequently, when calculating the pseudo-likelihood, the "effective" parameter vector only consists of $(\{\theta_{R1i}\}_{i=1}^{I}, \theta_c)$. When approximating the expected value function for each agent, we combine the common and individual-specific kernel weights to produce the weighted average of past pseudo-value functions.

The outer loop iteration for drawing a parameter vector from the posterior distribution can be broken down into three steps.

Step 1. Hyperparameter updating step: we draw the hyperparameter μ^r conditional on $\{\theta_{R1i}^{r-1}\}_{i=1}^{I}$. There is no inner loop associated with this step.
Step 2. Data augmentation step: we draw individual-specific parameters $\{\theta_{R1i}^{r}\}_{i=1}^{I}$ conditional on μ^r and θ_c^{r-1} using the M-H algorithm.
Step 3. Step for drawing common parameters: we draw θ_c^r using the M-H algorithm conditional on $\{\theta_{R1i}^{r}\}_{i=1}^{I}$.

Steps 1 and 2 are the additional steps compared to the base case where all parameters are common across agents. Also notice that the inner loops are nested within steps 2 and 3. The only difference between the two inner loops in steps 2 and 3 is that the value function will be solved conditional on a different parameter vector. In what follows, we will first describe the three steps in the outer loop, and then explain the inner loop. We assume that the prior is independent for (μ, θ_c), i.e., $\pi(\mu, \theta_c) = \pi(\mu) \cdot \pi(\theta_c)$.

(1) The outer loop
Suppose we are in iteration r with parameter estimates being $(\mu^r, \{\theta^r_{R1i}\}_{i=1}^I, \theta^r_c)$.

(1.1) Hyperparameter updating step
Given $\{\theta^{r-1}_{R1i}\}_{i=1}^I$, the posterior density for μ is proportional to

$$\pi(\mu) \prod_{i=1}^{I} g(\theta^{r-1}_{R1i}; \mu).$$

In most applications, we can draw μ^r directly from this posterior density.[14] Note that this step does not involve the solution of the DP problem.

(1.2) Data augmentation step
The model specification implies that $g(\theta_{R1i}; \mu^r)$ is effectively the prior for θ_{R1i}. For each agent i, we first draw a candidate parameter from the proposal density

$$\theta^{*r}_{R1i} \sim q(\theta^{*r}_{R1i} | \theta^{r-1}_{R1i}, \mu^r).$$

Then, accept θ^{*r}_{R1i} with probability λ. That is,

$$\theta^r_{R1i} = \begin{cases} \theta^{*r}_{R1i} & \text{with probability } \lambda, \\ \theta^{r-1}_{R1i} & \text{with probability } 1 - \lambda, \end{cases}$$

where

$$\lambda = \min \left\{ \frac{g(\theta^{*r}_{R1i}; \mu^r) \tilde{L}^r_i(\mathsf{a}_i | \mathsf{s}_i; \theta^{*r}_{R1i}, \theta^{r-1}_c) q(\theta^{r-1}_{R1i} | \theta^{*r}_{R1i}, \mu^r)}{g(\theta^{r-1}_{R1i}; \mu^r) \tilde{L}^r_i(\mathsf{a}_i | \mathsf{s}_i; \theta^{r-1}_{R1i}, \theta^{r-1}_c) q(\theta^{*r}_{R1i} | \theta^{r-1}_{R1i}, \mu^r)}, 1 \right\}.$$

Note that the computation of $\tilde{L}^r_i$ requires us to compute the pseudo-expected values for observations of agent i only, i.e., $\{\mathsf{a}_i, \mathsf{s}_i\}$. We will discuss how to compute this pseudo-expected value function in step 2 below.

(1.3) Step for drawing common parameters
This step is essentially the same as the outer loop step in the base case. We first draw a candidate parameter from the proposal density

$$\theta^{*r}_c \sim q(\theta^{*r}_c | \theta^{r-1}_c).$$

[14] In the example that we will discuss later, we assume g is a normal distribution and μ includes parameters for mean and standard deviation. Assuming that the prior on the mean parameters is normal and that for standard deviation parameters is inverse Wishart (or inverse Gamma if θ_{R1i} is a scalar), the posterior distribution for mean parameters is normal and that for standard deviation parameter is inverse Wishart. There are simple procedures for making a draw from both distributions (e.g., see Train 2003).

Then, accept θ_c^{*r} with probability λ. That is,

$$\theta_c^r = \begin{cases} \theta_c^{*r} & \text{with probability } \lambda, \\ \theta_c^{r-1} & \text{with probability } 1-\lambda, \end{cases}$$

where

$$\lambda = \min\left\{ \frac{\pi(\theta_c^{*r})\tilde{L}^r(\mathsf{a}|\mathsf{s}; \{\theta_{R1i}^r\}_{i=1}^I, \theta_c^{*r}) q(\theta_c^{r-1}|\theta_c^{*r})}{\pi(\theta_c^{r-1})\tilde{L}^r(\mathsf{a}|\mathsf{s}; \{\theta_{R1i}^r\}_{i=1}^I, \theta_c^{r-1}) q(\theta_c^{*r}|\theta_c^{r-1})}, 1 \right\}.$$

Note that the computation of $\tilde{L}^r$ requires us to compute the pseudo-expected values for all observations, i.e., $\{\mathsf{a}_i, \mathsf{s}_i\}_{i=1}^I$. Step 2 discusses how to compute them.

(2) The inner loop
Recall that in the current setting, the value function is individual-specific. This feature suggests that we need to store pseudo-value functions for each agent. Thus, in iteration r, the output of the algorithm is

$$H^r = \{\theta_c^{*l}, \{\theta_{R1i}^{*l}, \tilde{\mathcal{V}}(., \epsilon^l; \theta_{R1i}^{*l}, \theta_c^{*l})\}_{i=1}^I\}_{l=r-N}^{r-1}.$$

The pseudo-expected value function for agent i given (θ_{R1i}, θ_c) is then

$$\tilde{E}_{s',\epsilon'}^r[\mathcal{V}(s', \epsilon'; \theta_{R1i}, \theta_c)|s, a]$$
$$= \sum_{s' \in S}\left[\sum_{l=r-N}^{r-1} \tilde{\mathcal{V}}(s', \epsilon^l; \theta_{R1i}^{*l}, \theta_c^{*l}) \cdot \omega(\theta_{R1i}^{*l}, \theta_c^{*l}; \theta_{R1i}, \theta_c)\right] f(s'|s, a; \theta_s),$$

where

$$\omega(\theta_{R1i}^{*l}, \theta_c^{*l}; \theta_{R1i}, \theta_c) = \frac{K_h(\theta_{R1i}^{*l}, \theta_{R1i})K_h(\theta_c^{*l}, \theta_c)}{\sum_{k=r-N}^{r-1} K_h(\theta_{R1i}^{*k}, \theta_{R1i})K_h(\theta_c^{*k}, \theta_c)}.$$

With this pseudo-expected value function, we can construct the value of choosing each alternative conditional on (s, ϵ):

$$\tilde{V}_j^r(s, \epsilon_j; \theta_{R1i}, \theta_c) = R(s, j, \epsilon_j; \theta_{Ri}) + \beta \tilde{E}_{s',\epsilon'}^r[\mathcal{V}(s', \epsilon'; \theta_{R1i}, \theta_c)|s, a = j],$$

which can then be used to construct the pseudo-likelihood for agent i. We should emphasize that in step 1.2, we need to compute the pseudo-expected value function at $(\theta_{R1i}^{*r}, \theta_c^{r-1})$ while in step 1.3, we do so at $(\theta_{R1i}^r, \theta_c^{*r})$. In both steps, we use the same set of past pseudo-value functions.
To update the pseudo-value function for agent i, we can simply follow the base case: simulate a draw of ϵ^r, substitute it into $\tilde{V}_j^r$ for agent i above for all $s \in \mathcal{S}$, and obtain,

$$\tilde{\mathcal{V}}^r(s, \epsilon^r; \theta_{R1i}^{*r}, \theta_c^{*r}) = \max_{a \in A} \tilde{V}_a^r(s, \epsilon_a^r; \theta_{R1i}^{*r}, \theta_c^{*r}), \ \forall s \in \mathcal{S}.$$

3 A rewards program example

We now present an example and use it to illustrate how to apply the IJC algorithm to do Bayesian inference for a DDP model. Suppose that there are two supermarket chains in a city ($j = 1, 2$). Each supermarket chain offers a stamp card, which can be exchanged for a gift upon completion. The stamp card for a chain is valid for all stores in the same chain. Consumers get one stamp for each visit at any store of a chain with a purchase. Rewards programs at the two supermarket chains 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 that they complete the stamp card. Once consumers receive a gift, they will start with a blank stamp card again in the next period.

In each period, a consumer chooses which supermarket chain to visit. Each chain offers different prices for their products. Let p_{ijt} be the price index that consumer i faces in supermarket chain j at time t. We allow the price index to be individual specific to reflect that consumers may differ in terms of their consumption baskets (e.g., some consumers have babies and they need to shop for diapers, some consumers are vegetarian and they do not shop for meats, etc.). For simplicity, we assume that p_{ijt} is drawn from an *i.i.d.* normal distribution, $N(\bar{p}, \sigma_p^2)$, which is known to consumers, and they observe p_{ijt} in the period that they decide their choices.[15] Let $s_{ijt} \in \mathcal{S}_j \equiv \{0, 1, \ldots, \bar{S}_j - 1\}$ denote the number of stamps collected for chain j in period t before consumer i makes 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. The state space of this dynamic model is $\mathcal{S} \equiv \mathcal{S}_1 \times \mathcal{S}_2$.

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

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

where α_j captures the brand equity for chain j, γ is the price sensitivity, G_{ij} is consumer i's valuation of the gift for chain j, and ϵ_{ijt} is the *i.i.d.* idiosyncratic random utility term. We assume ϵ_{ijt} is unobserved to researchers and extreme-value distributed. G_{ij} is assumed to be normally distributed around G_j with the standard deviation, σ_{G_j}. We allow G_{ij} to differ across consumers to reflect that individual's valuation for a gift may vary.[16] In each period, consumers may choose not to go shopping ($j = 0$). The single period mean utility of no shopping is normalized to zero, i.e., $U_{i0t}(s_{it}, p_{it}) = \epsilon_{i0t}$.

[15]We will discuss how to estimate an extension where p_{ijt} is serially correlated in Section 4.4.

[16]Suppose that the gift is a vase. Some consumers may value it highly, but others who already have several vases at home, may not.

Consumer i's objective is to maximize the sum of the present discounted future utility:

$$\max_{\{b_{ijt}\}_{t=1}^{\infty}} E\left[\sum_{t=1}^{\infty}\beta^{t-1}\sum_{j=0}^{2} b_{ijt}U_{ijt}(s_{it}, p_{it})\right],$$

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

$$s_{ijt+1} = \begin{cases} s_{ijt}+1 & \text{if } s_{ijt} < \bar{S}_j - 1 \text{ and purchase at chain } j \text{ in period } t; \\ 0 & \text{if } s_{ijt} = \bar{S}_j - 1 \text{ and purchase at chain } j \text{ in period } t; \\ s_{ijt} & \text{if purchase at chain } -j \text{ or no shopping in period } t. \end{cases} \quad (7)$$

The parameters of the model are $\{\alpha_j, G_j, \sigma_{G_j}\}_{j=1}^{2}, \gamma, \beta, \bar{p}$, and σ_p. In what follows, we assume that $\bar{p}$ and σ_p are known to researchers. Note that this is not a strong assumption because these two parameters can be recovered from the price data alone. Let θ be the vector of parameters, and $G_i \equiv (G_{i1}, G_{i2})$.[17] Since the dynamic optimization problem is stationary, we drop the t subscript hereafter. The $E_\epsilon max$ function is given by: For each s_i, p_i,

$$\begin{aligned} \mathcal{W}(s_i, p_i; G_i, \theta) &\equiv E_\epsilon \max_{j\in\{0,1,2\}}\{W_j(s_i, p_i; G_i, \theta) + \epsilon_{ij}\} \\ &= \ln\left[\sum_{j=0}^{2}\exp(W_j(s_i, p_i; G_i, \theta))\right], \end{aligned} \quad (8)$$

where the second equality follows from the extreme value assumption on ϵ. Denote p' as the price vector next period. The alternative specific value functions are given by: For $j = 1, 2$,

$$\begin{aligned} &W_j(s_{ij}, s_{i-j}, p_i; G_i, \theta) \\ &\quad = \begin{cases} \alpha_j + \gamma p_{ij} + \beta E_{p'}[\mathcal{W}(s_{ij}+1, s_{i-j}, p'; G_i, \theta)] & \text{if } s_{ij} < \bar{S}_j - 1, \\ \alpha_j + \gamma p_{ij} + G_{ij} + \beta E_{p'}[\mathcal{W}(0, s_{i-j}, p'; G_i, \theta)] & \text{if } s_{ij} = \bar{S}_j - 1, \end{cases} \end{aligned} \quad (9)$$

$$W_0(s_i, p_i; G_i, \theta) = \beta E_{p'}[\mathcal{W}(s_i, p'; G_i, \theta)], \quad (10)$$

where $E_{p'}[\mathcal{W}(., p'; G_i, \theta)] = \int \mathcal{W}(., p'; G_i, \theta)dF(p')$ is the expected value function.

[17] With a slight abuse of notation, we use G_j to denote the mean value of the gift at store $j = 1, 2$, and $G_i = (G_{i1}, G_{i2})$ to denote the vector of the values of the gift for consumer i.

4 Estimation procedures

We now discuss how to estimate this model. We describe the conventional Bayesian estimation method and the IJC method in order, and highlight their differences.[18]

4.1 Conventional Bayesian approach (full-solution based method)

The conventional approach is essentially a nested fixed point algorithm proposed by Rust (1987). It proceeds in the following two main steps: the outer loop and inner loop. For simplicity, we present the procedure for the model without unobserved heterogeneity (i.e., $G_{ij} = G_j, \forall i$). We refer readers to Fig. 1 for a flowchart as they read the following two subsections.

4.1.1 The outer loop (MCMC algorithm)

Recall that the outer loop is a standard M-H algorithm. Let us use $(\mathsf{b}_{it}, \mathsf{s}_{it}, \mathsf{p}_{it})$, I and T_i to denote the observed data, total number of consumers, and total number of periods observed for each consumer i, respectively. Define $\mathsf{b} \equiv \{\mathsf{b}_{it}, \forall i, t\}$, $\mathsf{s} \equiv \{\mathsf{s}_{it}, \forall i, t\}$, $\mathsf{p} \equiv \{\mathsf{p}_{it}, \forall i, t\}$ and let $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta)$ be the likelihood of observing b:

$$L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta) = \prod_{i=1}^{I} \prod_{t=1}^{T_i} \prod_{j=0}^{2} \left(\frac{\exp(W_j(\mathsf{s}_{it}, \mathsf{p}_{it}; \theta))}{\sum_{k=0}^{2} \exp(W_k(\mathsf{s}_{it}, \mathsf{p}_{it}; \theta))} \right)^{\mathsf{b}_{ijt}} .$$

Let $\pi(\theta)$ be the prior on θ. The posterior density of θ is proportional to $\pi(\theta) \cdot L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta)$.

Suppose we are at iteration r with parameter estimates being θ^r. Then, we first draw the candidate parameter from the proposal density

$$\theta^{*r} \sim q(\theta^{*r}|\theta^{r-1}).$$

Then we accept θ^{*r}, i.e., set $\theta^r = \theta^{*r}$, with probability,

$$\lambda = \min \left\{ \frac{\pi(\theta^{*r}) L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r}) q(\theta^{r-1}|\theta^{*r})}{\pi(\theta^{r-1}) L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1}) q(\theta^{*r}|\theta^{r-1})}, 1 \right\};$$

and we reject θ^{*r}, i.e., set $\theta^r = \theta^{r-1}$, with probability $1 - \lambda$.

[18]For the identification issue of this model, see Ching et al. (2012).

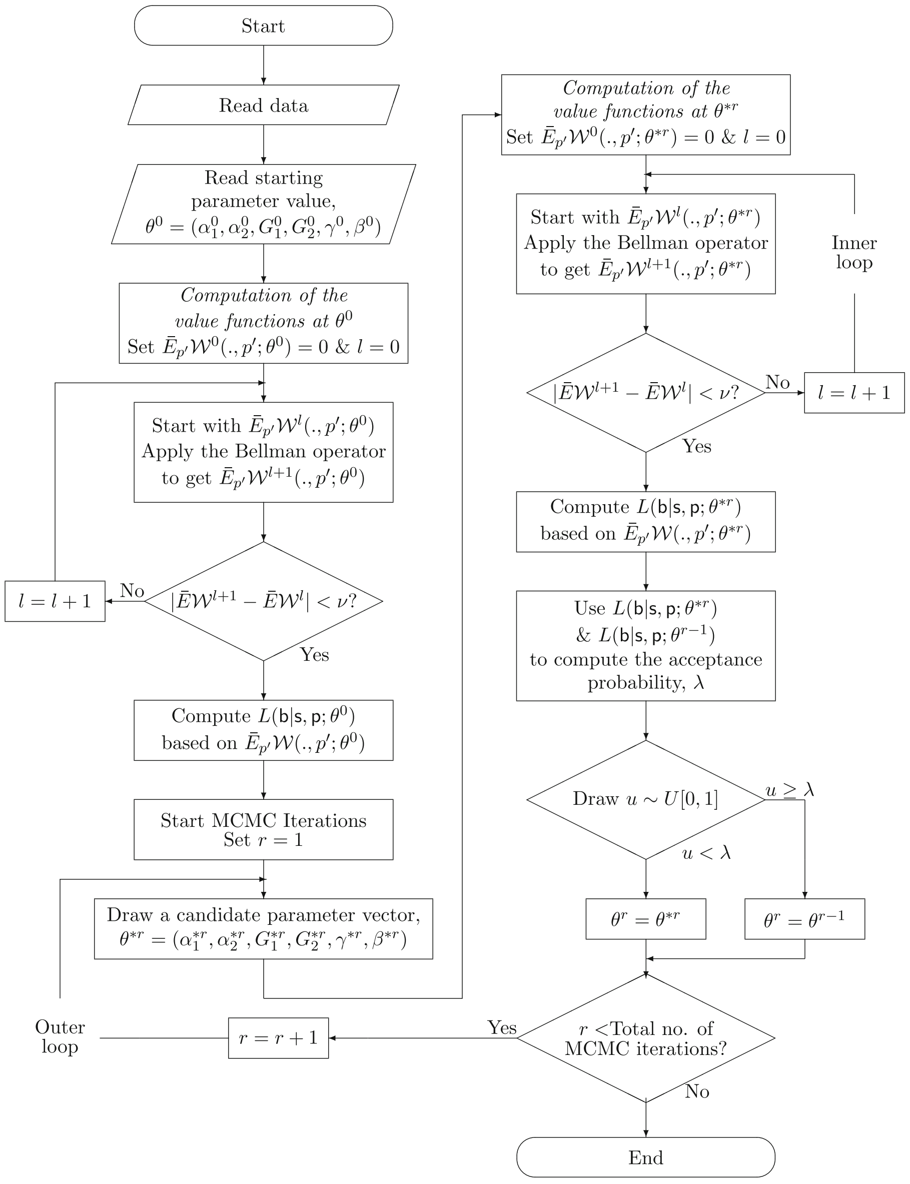
Start
Read data
Read starting parameter value, $\theta^0 = (\alpha_1^0, \alpha_2^0, G_1^0, G_2^0, \gamma^0, \beta^0)$
Computation of the value functions at θ^0
Set $\bar{E}_{p'}\mathcal{W}^0(., p'; \theta^0) = 0$ & $l = 0$
Start with $\bar{E}_{p'}\mathcal{W}^l(., p'; \theta^0)$
Apply the Bellman operator to get $\bar{E}_{p'}\mathcal{W}^{l+1}(., p'; \theta^0)$
$|\bar{E}\mathcal{W}^{l+1} - \bar{E}\mathcal{W}^l| < \nu$?
No
$l = l + 1$
Yes
Compute $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^0)$ based on $\bar{E}_{p'}\mathcal{W}(., p'; \theta^0)$
Start MCMC Iterations
Set $r = 1$
Draw a candidate parameter vector, $\theta^{*r} = (\alpha_1^{*r}, \alpha_2^{*r}, G_1^{*r}, G_2^{*r}, \gamma^{*r}, \beta^{*r})$
Outer loop
$r = r + 1$
Computation of the value functions at θ^{*r}
Set $\bar{E}_{p'}\mathcal{W}^0(., p'; \theta^{*r}) = 0$ & $l = 0$
Start with $\bar{E}_{p'}\mathcal{W}^l(., p'; \theta^{*r})$
Apply the Bellman operator to get $\bar{E}_{p'}\mathcal{W}^{l+1}(., p'; \theta^{*r})$
Inner loop
$|\bar{E}\mathcal{W}^{l+1} - \bar{E}\mathcal{W}^l| < \nu$?
No
$l = l + 1$
Yes
Compute $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r})$ based on $\bar{E}_{p'}\mathcal{W}(., p'; \theta^{*r})$
Use $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r})$ & $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ to compute the acceptance probability, λ
Draw $u \sim U[0, 1]$
$u \geq \lambda$
$u < \lambda$
$\theta^r = \theta^{*r}$
$\theta^r = \theta^{r-1}$
$r <$Total no. of MCMC iterations?
Yes
No
End

Fig. 1 Flowchart for the conventional Bayesian approach (homogeneous model)

Note that $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r})$ and $L(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ depend on the expected value function evaluated at θ^{*r} and θ^{r-1}, respectively. We now discuss how to implement *the inner loop*, which solves for the expected value functions numerically at these parameter vectors.

4.1.2 The inner loop (the method of successive approximation)

To solve the model described in Section 3 numerically, we take advantage of the contraction mapping property of the Bellman operator and apply the method of successive approximation as follows.

1. For each $j = 1, 2$, make M independent draws of $\{\tilde{p}_j^m\}_{m=1}^M$ from the price distribution function, $N(\bar{p}, \sigma_p^2)$. We denote the draws to be $\mathcal{P}^M = \{\tilde{p}^m, m = 1, ..., M\}$, where $\tilde{p}^m = (\tilde{p}_1^m, \tilde{p}_2^m)$, and fix them below.
2. Start with an arbitrary initial guess of $E_\epsilon max$ function, e.g., set $\mathcal{W}^0(s, \tilde{p}^m; \theta) = 0$, $\forall s \in \mathcal{S}$ and $\forall \tilde{p}^m$. Suppose that we know $\mathcal{W}^l$, where l indexes the number of iterations. Steps 3 & 4 discuss how to obtain $\mathcal{W}^{l+1}$.
3. For each s, substitute $\{\tilde{p}^m\}_{m=1}^M$ into $\mathcal{W}^l(s, p; \theta)$, and then take the average across $\tilde{p}^m$'s to obtain a Monte Carlo approximation of the expected value function:
$$\bar{E}_{p'}\mathcal{W}^l(s, p'; \theta) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{W}^l(s, \tilde{p}^m; \theta).$$
4. Substitute this approximated expected value function into the Bellman operator (i.e., Eqs. 8–10) and obtain $\mathcal{W}^{l+1}(s, \tilde{p}^m; \theta)$, that is, $\forall s \in \mathcal{S}, \forall \tilde{p}^m \in \mathcal{P}^M$,
$$\mathcal{W}^{l+1}(s, \tilde{p}^m; \theta) = \ln\left[\sum_{j=0}^{2} \exp(W_j^l(s, \tilde{p}^m; \theta))\right], \tag{11}$$
where for $j = 1, 2$,
$$W_j^l(s, \tilde{p}^m; \theta) \\ = \begin{cases} \alpha_j + \gamma\tilde{p}_j^m + \beta\bar{E}_{p'}\mathcal{W}^l(s_j + 1, s_{-j}, p'; \theta) & \text{if } s_j < \bar{S}_j - 1, \\ \alpha_j + \gamma\tilde{p}_j^m + G_{ij} + \beta\bar{E}_{p'}\mathcal{W}^l(0, s_{-j}, p'; \theta) & \text{if } s_j = \bar{S}_j - 1, \end{cases} \tag{12}$$
$$W_0^l(s, \tilde{p}^m; \theta) = \beta\bar{E}_{p'}\mathcal{W}^l(s, p'; \theta). \tag{13}$$
5. Repeat steps 3 and 4 until $\bar{E}_{p'}\mathcal{W}^{l+1}(., p'; \theta)$ converges. The convergence is determined by checking whether $\max_{s\in\mathcal{S}} ||\bar{E}_{p'}\mathcal{W}^{l+1}(s, p'; \theta) - \bar{E}_{p'}\mathcal{W}^l(s, p'; \theta)|| < \nu$, where ν is the tolerance level set by the researcher. Typically, the tolerance level is set at 1e–6.

In general, the computational burden increases exponentially with the number of state variables, and linearly with the number of values in each state variable. Also, the number of iterations required to achieve the convergence of $\bar{E}_{p'}\mathcal{W}^l(s, p'; \theta)$ increases as the discount factor β increases (e.g., see Santos and Rust 2004). We will now turn to discuss how to apply the IJC algorithm to estimate this rewards program model.

4.2 How to apply the IJC algorithm

By now, it should be clear that the main obstacle of the conventional Bayesian approach is the computational burden of solving for the expected value function at a large number of parameter vectors simulated from the M-H algorithm. As discussed earlier, the main difference between the IJC and conventional Bayesian methods is the inner loop. Instead of using the contraction mapping argument to obtain the expected value function, the IJC method uses the past pseudo-$E_\epsilon max$ functions (or pseudo-value functions in the general case) to form a non-parametric estimate of the expected value function evaluated at the current draw of parameter vector. More precisely, the non-parametric estimate is a weighted average of the pseudo-$E_\epsilon max$ functions obtained as past outcomes of the estimation algorithm. The weights depend on the distance between the past parameter vector draws and the current one – the shorter the distance, the higher the weight. Later, we will provide evidence that such a non-parametric estimate is usually computationally much cheaper than the method of successive approximation described in Section 4.1.2, especially for β close to 1. Consequently, the IJC algorithm has the potential to significantly reduce the computational burden per iteration compared with the conventional Bayesian approach.

For simplicity, suppose that there is no unobserved consumer heterogeneity (i.e., $G_{ij} = G_j, \forall i$). The output of the IJC algorithm in each iteration r is $\{\theta^r, \theta^{*r}, \tilde{\mathcal{W}}^r(., \tilde{p}^r; \theta^{*r})\}$, where $\tilde{p}^r_j$ is a draw from $N(\bar{p}, \sigma_p^2)$ for $j = 1, 2$, and $\tilde{\mathcal{W}}^r(., \tilde{p}^r; \theta^{*r})$ is the pseudo-$E_\epsilon max$ function of s, given $\tilde{p}^r$ and θ^{*r}. We store $H^r = \{\theta^{*l}, \tilde{\mathcal{W}}^l(., \tilde{p}^l; \theta^{*l})\}_{l=r-N}^{r-1}$. It should be emphasized that $\tilde{\mathcal{W}}^l(., \tilde{p}^l; \theta^{*l})$ refers to $\tilde{\mathcal{W}}^l(s, \tilde{p}^l; \theta^{*l})$ for all $s \in \mathcal{S}$. In other words, we need to store $\tilde{\mathcal{W}}^l$ at all s. The pseudo-$E_\epsilon max$ functions, $\tilde{\mathcal{W}}^r$, and the pseudo-alternative specific value functions, $\tilde{W}^r_j$, are defined as follows. First, we draw $\tilde{p}^r_j$ from $N(\bar{p}, \sigma_p^2)$, for $j = 1, 2$.[19] Then, for each $s \in \mathcal{S}$,

$$\tilde{\mathcal{W}}^r(s, \tilde{p}^r; \theta^{*r}) = \ln\left[\sum_{j=0}^{2} \exp(\tilde{W}^r_j(s, \tilde{p}^r; \theta^{*r}))\right], \tag{14}$$

where for $j = 1, 2$,

$$\tilde{W}^r_j(s, \tilde{p}^r; \theta^{*r}) = \begin{cases} \alpha_j + \gamma \tilde{p}^r_j + \beta \tilde{E}^r_{p'} \mathcal{W}(s_j + 1, s_{-j}, p'; \theta^{*r}) & \text{if } s_j < \bar{S}_j - 1, \\ \alpha_j + \gamma \tilde{p}^r_j + G_j + \beta \tilde{E}^r_{p'} \mathcal{W}(0, s_{-j}, p'; \theta^{*r}) & \text{if } s_j = \bar{S}_j - 1, \end{cases} \tag{15}$$

$$\tilde{W}^r_0(s; \tilde{p}^r; \theta^{*r}) = \beta \tilde{E}^r_{p'} \mathcal{W}(s, p'; \theta^{*r}). \tag{16}$$

[19]Here we propose to make one draw of price vector in each iteration. However, in practice, we find it useful to draw several price vectors in each iteration and store the average of pseudo-$E_\epsilon max$ functions evaluated at these draws of price vectors. We will discuss this procedure in Appendix A.

The *pseudo*-expected value function, $\tilde{E}^r_{p'}\mathcal{W}(., p'; \theta^{*r})$, is defined as the weighted average of the past pseudo-$E_\epsilon max$ functions obtained from the estimation algorithm. For instance, $\tilde{E}^r_{p'}\mathcal{W}(s, p'; \theta^{*r})$ can be constructed as follows:

$$\tilde{E}^r_{p'}\mathcal{W}(s, p'; \theta^{*r}) = \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; \theta^{*l}) \frac{K_h(\theta^{*l}, \theta^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta^{*k}, \theta^{*r})}, \tag{17}$$

where $K_h(.)$ is a Gaussian kernel with bandwidth h. Note that the price shock is integrated out by this weighted average of the past pseudo-$E_\epsilon max$ functions evaluated at random draws of $\{\tilde{p}^l\}_{l=r-N}^{r-1}$.

4.3 Details of the IJC algorithm

Now we provide a step-by-step guide to implement the IJC algorithm. The steps are similar to the conventional Bayesian approach, except that we use Eq. 17 to approximate the expected value function. Once the readers understand how to implement the algorithm in this simple example, they should be able to extend it to more complicated settings. We consider two versions of the model: (i) without unobserved consumer heterogeneity, and (ii) with unobserved consumer heterogeneity.

4.3.1 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 $j = 1, 2$). To assist readers to understand these steps, we summarize a list of notations in Table 1

Table 1 List of notations for homogeneous model in Section 4.3.1

Notation	Description
θ^{*r}	Candidate parameter vector in iteration r
θ^r	Accepted parameter vector in iteration r
$\tilde{p}^r$	$= (\tilde{p}^r_1, \tilde{p}^r_2)$; a draw of price vector in iteration r such that $\tilde{p}^r_j \sim N(\bar{p}, \sigma^2_p)$
$\tilde{W}^r_j(s, \tilde{p}^r; \theta^{*r})$	Pseudo-alternative specific value function for alternative j in iteration r conditional on $(s, \tilde{p}^r; \theta^{*r})$
$\tilde{\mathcal{W}}^r(s, \tilde{p}^r; \theta^{*r})$	$= E_\epsilon \max_j\{\tilde{W}^r_j(s, \tilde{p}^r; \theta^{*r}) + \epsilon_{ij}\}$; pseudo-$E_\epsilon max$ function in iteration r conditional on $(s, \tilde{p}^r; \theta^{*r})$
H^r	$= \{\theta^{*l}, \tilde{\mathcal{W}}^l(., \tilde{p}^l; \theta^{*l})\}_{l=r-N}^{r-1}$; set of past pseudo-$E_\epsilon max$ functions used for approximating the expected value functions in iteration r
$\tilde{E}^r_{p'}\mathcal{W}(s, p'; \theta^{*r})$	$= \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; \theta^{*l}) \frac{K_h(\theta^{*l}, \theta^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta^{*k}, \theta^{*r})}$; pseudo-expected value function in iteration r conditional on $(s; \theta^{*r})$
b	$= \{\mathsf{b}_{ijt}\ \forall i, j, t\}$; a vector of observed buying decisions
$\tilde{L}^r(\mathsf{b}\|\mathsf{s}, \mathsf{p}; \theta^{*r})$	Pseudo-likelihood conditional on H^r, (s, p), and θ^{*r}

and create a flowchart in Fig. 2; readers may refer to them as they read through the steps below.

1. Suppose that we are at iteration r. Let N be the number of past pseudo-$E_\epsilon max$ functions that we plan to store. More precisely, for $r \geq N$, we store,

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

   We will discuss how to modify the steps when $r < N$ later.

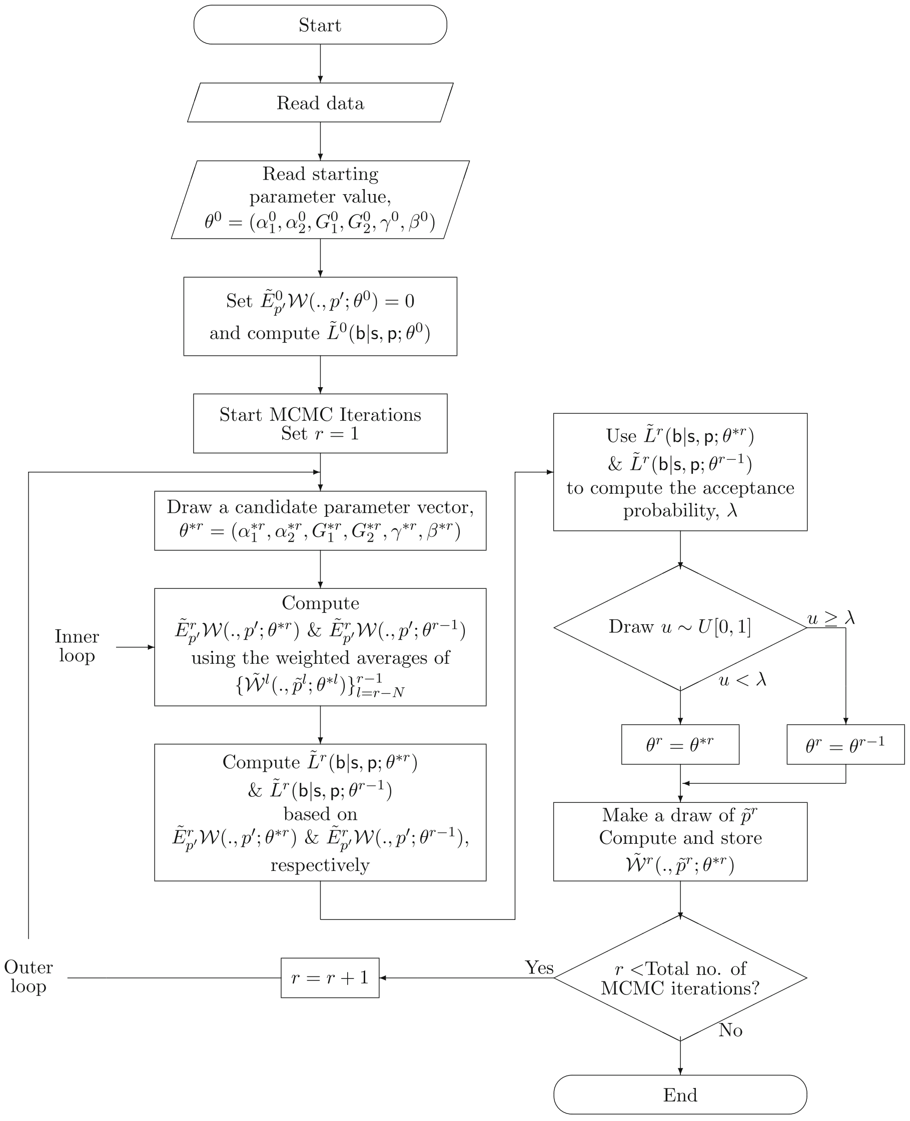
Start
Read data
Read starting parameter value, $\theta^0 = (\alpha_1^0, \alpha_2^0, G_1^0, G_2^0, \gamma^0, \beta^0)$
Set $\tilde{E}^0_{p'}\mathcal{W}(., p'; \theta^0) = 0$ and compute $\tilde{L}^0(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^0)$
Start MCMC Iterations Set $r = 1$
Draw a candidate parameter vector, $\theta^{*r} = (\alpha_1^{*r}, \alpha_2^{*r}, G_1^{*r}, G_2^{*r}, \gamma^{*r}, \beta^{*r})$
Inner loop
Compute $\tilde{E}^r_{p'}\mathcal{W}(., p'; \theta^{*r})$ & $\tilde{E}^r_{p'}\mathcal{W}(., p'; \theta^{r-1})$ using the weighted averages of $\{\tilde{\mathcal{W}}^l(., \tilde{p}^l; \theta^{*l})\}_{l=r-N}^{r-1}$
Compute $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r})$ & $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ based on $\tilde{E}^r_{p'}\mathcal{W}(., p'; \theta^{*r})$ & $\tilde{E}^r_{p'}\mathcal{W}(., p'; \theta^{r-1})$, respectively
Use $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*r})$ & $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ to compute the acceptance probability, λ
Draw $u \sim U[0,1]$
$u \geq \lambda$
$u < \lambda$
$\theta^r = \theta^{*r}$
$\theta^r = \theta^{r-1}$
Make a draw of $\tilde{p}^r$ Compute and store $\tilde{\mathcal{W}}^r(., \tilde{p}^r; \theta^{*r})$
$r <$Total no. of MCMC iterations?
Yes
No
$r = r + 1$
Outer loop
End

Fig. 2 Flowchart for the IJC algorithm (homogeneous model)

2. Draw a candidate parameter vector, θ^{*r}, from a proposal distribution $q(\theta^{*r}|\theta^{r-1})$.
3. Compute the pseudo-likelihood conditional on θ^{*r}, $\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{*r})$,

$$\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{*r}) = \prod_{i=1}^{I}\prod_{t=1}^{T_i}\prod_{j=0}^{2}\left(\frac{\exp(\tilde{W}_j^r(\mathsf{s}_{it},\mathsf{p}_{it};\theta^{*r}))}{\sum_{k=0}^{2}\exp(\tilde{W}_k^r(\mathsf{s}_{it},\mathsf{p}_{it};\theta^{*r}))}\right)^{\mathsf{b}_{ijt}}.$$

   To obtain $\tilde{W}_j^r$, we need to calculate $\tilde{E}_{p'}^r\mathcal{W}(.,p';\theta^{*r})$, which is obtained using the weighted average of the past pseudo-$E_\epsilon max$ functions:$\{\tilde{\mathcal{W}}^l(.,\tilde{p}^l;\theta^{*l})\}_{l=r-N}^{r-1}$. The weights are determined by Gaussian independent kernels with bandwidth h. For all s,

$$\tilde{E}_{p'}^r\mathcal{W}(s,p';\theta^{*r}) = \sum_{l=r-N}^{r-1}\tilde{\mathcal{W}}^l(s,\tilde{p}^l;\theta^{*l})\frac{K_h(\theta^{*l},\theta^{*r})}{\sum_{k=r-N}^{r-1}K_h(\theta^{*k},\theta^{*r})}.$$

   Substituting this into Eqs. 15 and 16 gives us $\tilde{W}_j^r$.
   Similarly, compute the pseudo-likelihood conditional on θ^{r-1}, $\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{r-1})$. Let $\pi(\cdot)$ be the prior distribution of the parameter vector. Then we determine whether or not to accept θ^{*r} based on the acceptance probability,

$$\min\left\{\frac{\pi(\theta^{*r})\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{*r})q(\theta^{r-1}|\theta^{*r})}{\pi(\theta^{r-1})\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{r-1})q(\theta^{*r}|\theta^{r-1})},1\right\}.$$

   If accept, set $\theta^r = \theta^{*r}$; otherwise, set $\theta^r = \theta^{r-1}$.
4. Since we propose to store the pseudo-$E_\epsilon max$ functions at θ^{*r}, we need to compute $\tilde{\mathcal{W}}^r(.,\tilde{p}^r;\theta^{*r})$.
   (a) Make one draw of $\tilde{p}_j^r$ from $N(\bar{p},\sigma_p^2)$, for $j=1,2$.
   (b) Compute $\tilde{W}_j^r(.,\tilde{p}^r;\theta^{*r})$ for $j=0,1,2$, using $\tilde{E}_{p'}^r\mathcal{W}(.,p';\theta^{*r})$ obtained from step 3.
   (c) Given $\tilde{W}_j^r(.,\tilde{p}^r;\theta^{*r})$ for $j=0,1,2$, obtain the pseudo-$E_\epsilon max$ function, $\tilde{\mathcal{W}}^r(.,\tilde{p}^r;\theta^{*r})$, using Eq. 14.
   (d) Update H^{r+1}.
5. Go to iteration $r+1$.

We make three remarks here. First, when we start the algorithm (i.e., $r=1$), H^1 is empty, and we set $\tilde{E}_{p'}^1\mathcal{W}(.,p';\theta^0) = \tilde{E}_{p'}^1\mathcal{W}(.,p';\theta^{*1}) = 0$. Second, for $r < N$, we set $H^r = \{\theta^{*l},\tilde{\mathcal{W}}^l(.,\tilde{p}^l;\theta^{*l})\}_{l=1}^{r-1}$, and the calculation of $\tilde{E}_{p'}^r\mathcal{W}(.,p';\theta^{*r})$ should be modified accordingly (i.e., the summation should start from $l=1$ to $r-1$ instead of from $l=r-N$ to $r-1$). Third, in step 3, we need to compute the pseudo-likelihood at previously accepted parameter vector, θ^{r-1}. It may first seem that this is redundant because the pseudo-likelihood at θ^{r-1} has been computed in the previous iteration, and thus we can re-use it in the current iteration. This is true in a full-solution based Bayesian MCMC algorithm, where we solve for the $E_\epsilon max$ function exactly in each iteration. However, in the IJC algorithm, the likelihood in iteration r depends on H^r (this is why we call it pseudo-likelihood), which is updated in each iteration. Thus, in principle, we need to compute the pseudo-likelihood at θ^{r-1} using the updated set of past pseudo-$E_\epsilon max$ functions in iteration r.[20]

4.3.2 Heterogeneous consumers

We now present the implementation of the IJC algorithm when consumers have heterogeneous valuations for the reward (i.e., $\sigma_{G_j} > 0$).

We will take the Hierarchical Bayes approach and treat G_{ij} as individual-specific parameters in our estimation. In this case, the parameter vector can be partitioned into three parts with $\theta = (\mu, \{G_i\}_{i=1}^I, \theta_c)$, where $\mu = (G_1, G_2, \sigma_{G_1}, \sigma_{G_2})$; $G_i = (G_{i1}, G_{i2})$; $\theta_c = (\alpha_1, \alpha_2, \gamma, \beta)$. One can treat μ as the vector of hyperparameters that captures the distribution of the individual-specific parameters, and θ_c consists of parameters that are common across consumers.[21] To simplify the discussion, we use a normal prior on G_j and an inverted gamma prior on σ_{G_j}. We assume that they are independent across j. The prior on θ_c can be flexible and we use independent flat priors in our estimation exercise.

Each MCMC iteration mainly consists of three blocks.

(i) Draw μ^r, that is, for $j = 1, 2$, draw $G_j^r \sim f_G(.|\sigma_{G_j}^{r-1}, \{G_{ij}^{r-1}\}_{i=1}^I)$ and $\sigma_{G_j}^r \sim f_\sigma(.|G_j^r, \{G_{ij}^{r-1}\}_{i=1}^I)$ where f_G and f_σ are the conditional posterior distributions.

(ii) Draw individual parameters $G_i^r \sim f_i(.|\mathsf{b}_i, \mathsf{s}_i, \mathsf{p}_i, \mu^r, \theta_c^{r-1})$ using the M-H algorithm.

(iii) Draw $\theta_c^r \sim f_{\theta_c}(.|\mathsf{b}, \mathsf{s}, \mathsf{p}, \{G_i^r\}_{i=1}^I)$ using the M-H algorithm. Note that this block is similar to the steps described in the homogeneous case.

Heterogeneity introduces an additional complication that the expected value functions need to be approximated for each consumer. As before, this is achieved by taking a weighted average of past pseudo-$E_\epsilon max$ functions based on the distance of the current parameter vector to the past parameter vector. It should be emphasized that conditional on $\{G_i\}_{i=1}^I$, the expected value functions (and hence the likelihood functions) do not depend on μ. Consequently, when calculating the likelihoods in blocks (ii) and (iii), the "effective" parameter vector only consists of $(\{G_i\}_{i=1}^I, \theta_c)$.

We now describe the details of the estimation steps. To assist the readers to follow these steps, we summarize a list of notations in Table 2 and provide a flowchart in Fig. 3; readers may refer to them as they read through the steps

[20] In practice, however, it may not be worthwhile to compute the pseudo-likelihood at θ^{r-1} in every iteration because the set of past pseudo-$E_\epsilon max$ functions is updated by only one element in each iteration. Therefore, the pseudo-likelihood based on H^{r-1} could be a good approximation for the pseudo-likelihood based on H^r. We will discuss more details in Appendix A.

[21] In terms of the notation in Section 2.5, $\mu = \mu$, $G_i = \theta_{R1i}$, and $\theta_c = \theta_{R2}$.

Table 2 List of notations for heterogeneous model in Section 4.3.2

Notation	Description
G_j^r	Draw of G_j (population mean) in iteration r
$\sigma_{G_j}^r$	Draw of σ_{G_j} (population standard deviation) in iteration r
G_i^{*r}	$= (G_{i1}^{*r}, G_{i2}^{*r})$; Candidate parameter value specific to consumer i in iteration r
G_i^r	$= (G_{i1}^r, G_{i2}^r)$; Accepted parameter value specific to consumer i in iteration r
θ_c^{*r}	Candidate parameter vector common across consumers in iteration r
θ_c^r	Accepted parameter vector common across consumers in iteration r
$\tilde{p}^r$	$= (\tilde{p}_1^r, \tilde{p}_2^r)$; a draw of price vector in iteration r such that $\tilde{p}_j^r \sim N(\bar{p}, \sigma_p^2)$
$\tilde{W}_j^r(s, \tilde{p}^r; G_i^{*r}, \theta_c^{*r})$	Consumer i's pseudo-alternative specific value function for alternative j in iteration r conditional on $(s, \tilde{p}^r; G_i^{*r}, \theta_c^{*r})$
$\tilde{\mathcal{W}}^r(s, \tilde{p}^r; G_i^{*r}, \theta_c^{*r})$	$= E_\epsilon \max_j \{\tilde{W}_j^r(s, \tilde{p}^r; G_i^{*r}, \theta_c^{*r}) + \epsilon_{ij}\}$; consumer i's pseudo-$E_\epsilon max$ function in iteration r conditional on $(s, \tilde{p}^r; G_i^{*r}, \theta_c^{*r})$
H^r	$= \{\theta_c^{*l}, \{G_i^{*r}, \tilde{\mathcal{W}}^l(., \tilde{p}^l; G_i^{*r}, \theta_c^{*l})\}_{i=1}^I\}_{l=r-N}^{r-1}$; set of past pseudo-$E_\epsilon max$ functions used for approximating the expected future value functions in iteration r
$\tilde{E}_{p'}^r \mathcal{W}(s, p'; G_i^{*r}, \theta_c^{*r})$	$= \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_i^{*r}, \theta_c^{*l}) \frac{K_h(\theta_c^{*l}, \theta_c^{*r}) K_h(G_i^{*l}, G_i^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta_c^{*k}, \theta_c^{*r}) K_h(G_i^{*k}, G_i^{*r})}$; pseudo-expected value function for consumer i in iteration r conditional on $(s; G_i^{*r}, \theta_c^{*r})$
b_i	$= \{\mathsf{b}_{ijt}\ \forall j, t\}$; a vector of observed buying decisions for consumer i
$\tilde{L}_i^r(\mathsf{b}_i \mid \mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{*r})$	Pseudo-likelihood for consumer i conditional on H^r, $(\mathsf{s}_i, \mathsf{p}_i)$, and $(G_i^{*r}, \theta_c^{*r})$
b	$= \{\mathsf{b}_{ijt}\ \forall i, j, t\}$; a vector of observed buying decisions
$\tilde{L}^r(\mathsf{b} \mid \mathsf{s}, \mathsf{p}; \{G_i^{*r}\}_{i=1}^I, \theta_c^{*r})$	Joint pseudo-likelihood conditional on H^r, (s, p), and $(\{G_i^{*r}\}_{i=1}^I, \theta_c^{*r})$

below. Steps 2 and 3 belong to block (i), step 4 belongs to block (ii) and step 5 belongs to block (iii). Note that we only describe steps 2 and 3 briefly here because they are standard. For the details of these two steps, we refer readers to Chapter 12 of Train (2003).

1. Suppose that we are at iteration r. We start with

$$H^r = \{\theta_c^{*l}, \{G_i^{*l}, \tilde{\mathcal{W}}^l(., \tilde{p}^l; G_i^{*l}, \theta_c^{*l})\}_{i=1}^I\}_{l=r-N}^{r-1},$$

   where I is the number of consumers; N is the number of past iterations used for the expected value function approximation.
2. For each $j = 1, 2$, draw G_j^r (population mean of G_{ij}) from the posterior density conditional on $\sigma_{G_j}^{r-1}$ and $\{G_{ij}^{r-1}\}_{i=1}^I$.
3. For each $j = 1, 2$, draw $\sigma_{G_j}^r$ (population variance of G_{ij}) from the posterior density conditional on G_j^r and $\{G_{ij}^{r-1}\}_{i=1}^I$.
   - Steps 2 and 3 give us $\mu^r = (G_1^r, G_2^r, \sigma_{G_1}^r, \sigma_{G_2}^r)$.
4. For each $i = 1, ..., I$, draw G_i^r from its posterior distribution conditional on $(\mathsf{b}_i, \mathsf{s}_i, \mathsf{p}_i, \mu^r, \theta_c^{r-1})$, $f_i(G_i \mid \mathsf{b}_i, \mathsf{s}_i, \mathsf{p}_i, \mu^r, \theta_c^{r-1})$. Recall that our model specification implies that the "effective" prior on G_{ij} is $N(G_j^r, (\sigma_{G_j}^r)^2)$.

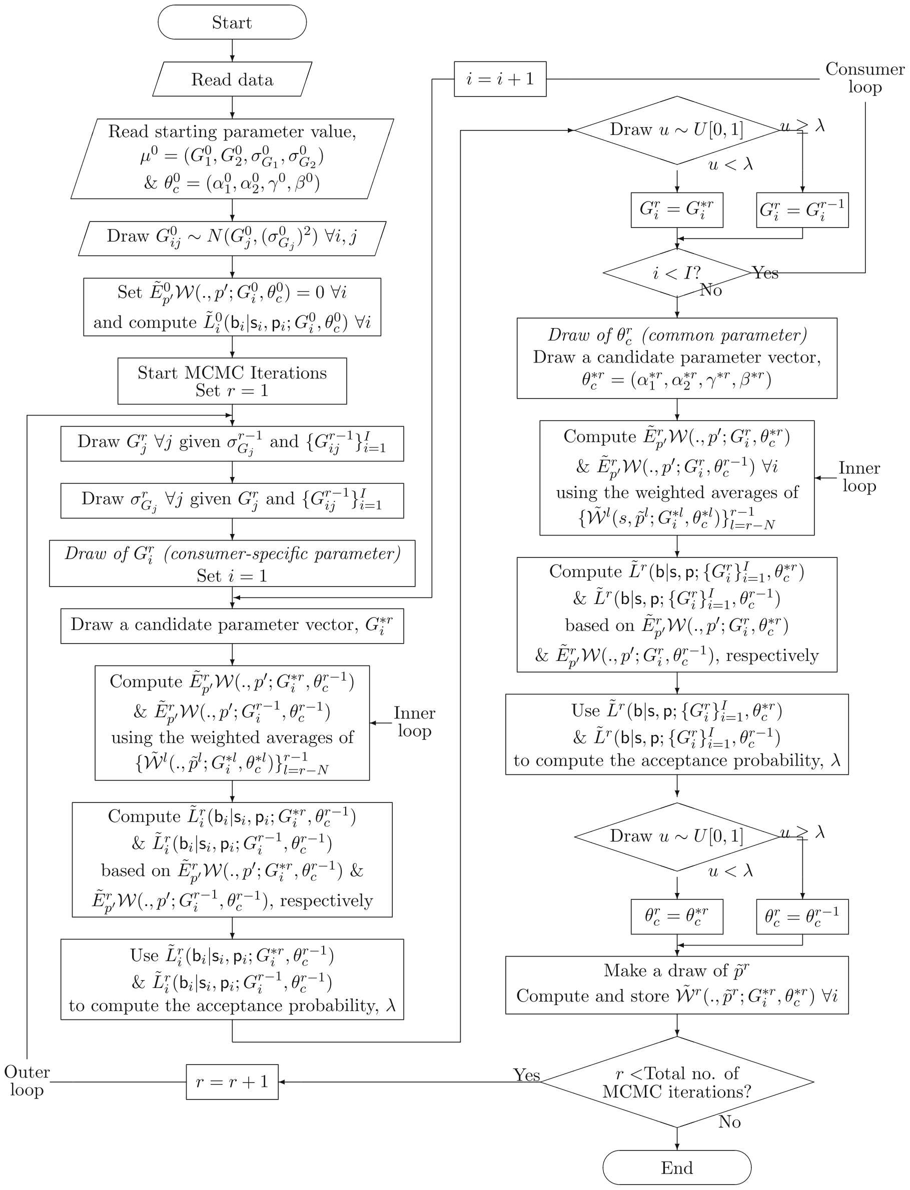
Start
Read data
Read starting parameter value, $\mu^0 = (G_1^0, G_2^0, \sigma_{G_1}^0, \sigma_{G_2}^0)$ & $\theta_c^0 = (\alpha_1^0, \alpha_2^0, \gamma^0, \beta^0)$
Draw $G_{ij}^0 \sim N(G_j^0, (\sigma_{G_j}^0)^2)\ \forall i, j$
Set $\tilde{E}_{p'}^0 \mathcal{W}(., p'; G_i^0, \theta_c^0) = 0\ \forall i$ and compute $\tilde{L}_i^0(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^0, \theta_c^0)\ \forall i$
Start MCMC Iterations Set $r = 1$
Draw $G_j^r\ \forall j$ given $\sigma_{G_j}^{r-1}$ and $\{G_{ij}^{r-1}\}_{i=1}^I$
Draw $\sigma_{G_j}^r\ \forall j$ given G_j^r and $\{G_{ij}^{r-1}\}_{i=1}^I$
Draw of G_i^r (consumer-specific parameter) Set $i = 1$
Draw a candidate parameter vector, G_i^{*r}
Compute $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^{*r}, \theta_c^{r-1})$ & $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^{r-1}, \theta_c^{r-1})$ using the weighted averages of $\{\tilde{\mathcal{W}}^l(., \tilde{p}^l; G_i^{*l}, \theta_c^{*l})\}_{l=r-N}^{r-1}$
Inner loop
Compute $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{r-1})$ & $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{r-1}, \theta_c^{r-1})$ based on $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^{*r}, \theta_c^{r-1})$ & $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^{r-1}, \theta_c^{r-1})$, respectively
Use $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{r-1})$ & $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{r-1}, \theta_c^{r-1})$ to compute the acceptance probability, λ
Outer loop
$r = r + 1$
$i = i + 1$
Consumer loop
Draw $u \sim U[0,1]$
$u \geq \lambda$
$u < \lambda$
$G_i^r = G_i^{*r}$
$G_i^r = G_i^{r-1}$
$i < I$?
Yes
No
Draw of θ_c^r (common parameter) Draw a candidate parameter vector, $\theta_c^{*r} = (\alpha_1^{*r}, \alpha_2^{*r}, \gamma^{*r}, \beta^{*r})$
Compute $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^r, \theta_c^{*r})$ & $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^r, \theta_c^{r-1})\ \forall i$ using the weighted averages of $\{\tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_i^{*l}, \theta_c^{*l})\}_{l=r-N}^{r-1}$
Inner loop
Compute $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r})$ & $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{r-1})$ based on $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^r, \theta_c^{*r})$ & $\tilde{E}_{p'}^r \mathcal{W}(., p'; G_i^r, \theta_c^{r-1})$, respectively
Use $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r})$ & $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{r-1})$ to compute the acceptance probability, λ
Draw $u \sim U[0,1]$
$u \geq \lambda$
$u < \lambda$
$\theta_c^r = \theta_c^{*r}$
$\theta_c^r = \theta_c^{r-1}$
Make a draw of $\tilde{p}^r$ Compute and store $\tilde{\mathcal{W}}^r(., \tilde{p}^r; G_i^{*r}, \theta_c^{*r})\ \forall i$
$r <$Total no. of MCMC iterations?
Yes
No
End

Fig. 3 Flowchart for the IJC algorithm (heterogeneous model)

Therefore, $f_i(G_i|\mathsf{b}_i, \mathsf{s}_i, \mathsf{p}_i, \mu^r, \theta_c^{r-1}) \propto \phi(G_i|\mu^r)\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i, \theta_c^{r-1})$, where $\phi(G_i|\mu^r)$ is the normal density. Since there is no easy way to draw from this posterior, we use the M-H algorithm. We use G_i^{*r} to denote the candidate value for G_i^r.

(a) Draw G_i^{*r} from a proposal distribution $q(G_i^{*r}|G_i^{r-1}, \mu^r)$.

(b) Compute the pseudo-likelihood for consumer i at G_i^{*r}, i.e., $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{r-1})$. The pseudo-likelihood is expressed as

$$\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{r-1}) = \prod_{t=1}^{T_i}\prod_{j=0}^{2}\left(\frac{\exp(\tilde{W}_j^r(\mathsf{s}_{it}, \mathsf{p}_{it}; G_i^{*r}, \theta_c^{r-1}))}{\sum_{k=0}^{2}\exp(\tilde{W}_k^r(\mathsf{s}_{it}, \mathsf{p}_{it}; G_i^{*r}, \theta_c^{r-1}))}\right)^{\mathsf{b}_{ijt}}.$$

To obtain $\tilde{W}_j^r$, we need $\tilde{E}_{p'}^r\mathcal{W}(., p'; G_i^{*r}, \theta_c^{r-1})$, which is obtained by a weighted average of past pseudo-$E_\epsilon max$ functions, $\{\tilde{\mathcal{W}}^l(., \tilde{p}^l; G_i^{*l}, \theta_c^{*l})\}_{l=r-N}^{r-1}$, treating G_i as one of the parameters when computing the weights. In the case of independent kernels, for all s,

$$\tilde{E}_{p'}^r\mathcal{W}(s, p'; G_i^{*r}, \theta_c^{r-1}) = \sum_{l=r-N}^{r-1}\tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_i^{*l}, \theta_c^{*l}) \times \frac{K_h(\theta_c^{*l}, \theta_c^{r-1})K_h(G_i^{*l}, G_i^{*r})}{\sum_{k=r-N}^{r-1}K_h(\theta_c^{*k}, \theta_c^{r-1})K_h(G_i^{*k}, G_i^{*r})}.$$

We repeat the same step to obtain the pseudo-likelihood conditional on $(G_i^{r-1}, \theta_c^{r-1})$, $\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{r-1}, \theta_c^{r-1})$. Then, we determine whether or not to accept G_i^{*r} with the acceptance probability, λ:

$$\lambda = \min\left\{\frac{\phi(G_i^{*r}|\mu^r)\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{*r}, \theta_c^{r-1})q(G_i^{r-1}|G_i^{*r}, \mu^r)}{\phi(G_i^{r-1}|\mu^r)\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i, \mathsf{p}_i; G_i^{r-1}, \theta_c^{r-1})q(G_i^{*r}|G_i^{r-1}, \mu^r)}, 1\right\}.$$

If accept, set $G_i^r = G_i^{*r}$; otherwise, set $G_i^r = G_i^{r-1}$.[22]

(c) Repeat (a) & (b) for all i.

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

(a) Draw $\theta_c^{*r} = (\alpha_1^{*r}, \alpha_2^{*r}, \gamma^{*r}, \beta^{*r})$ from a proposal distribution, $q(\theta_c^{*r}|\theta_c^{r-1})$.

(b) We then compute the pseudo-likelihood conditional on θ_c^{*r} and $\{G_i^r\}_{i=1}^I$, based on the pseudo-alternative specific value functions. The pseudo-likelihood, $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r})$, is expressed as

$$\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r}) = \prod_{i=1}^{I}\prod_{t=1}^{T_i}\prod_{j=0}^{2}\left(\frac{\exp(\tilde{W}_j^r(\mathsf{s}_{it}, \mathsf{p}_{it}; G_i^r, \theta_c^{*r}))}{\sum_{k=0}^{2}\exp(\tilde{W}_k^r(\mathsf{s}_{it}, \mathsf{p}_{it}; G_i^r, \theta_c^{*r}))}\right)^{\mathsf{b}_{ijt}}.$$

To obtain $\tilde{W}_j^r(., .; G_i^r, \theta_c^{*r}))$, we need to calculate $\tilde{E}_{p'}^r\mathcal{W}(., p'; G_i^r, \theta_c^{*r})$, which is a weighted average of the past pseudo-$E_\epsilon max$ functions,

[22] Note that if $q(., .)$ is symmetric, the expression of the acceptance probability will be simplified to $\lambda = \min\left\{\frac{\pi(G_i^{*r}|\mu^r)\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i,\mathsf{p}_i; G_i^{*r},\theta_c^{r-1})}{\pi(G_i^{r-1}|\mu^r)\tilde{L}_i^r(\mathsf{b}_i|\mathsf{s}_i,\mathsf{p}_i; G_i^{r-1},\theta_c^{r-1})}, 1\right\}$.

$\{\tilde{\mathcal{W}}^l(., \tilde{p}^l; G_i^{*l}, \theta_c^{*l})\}_{l=r-N}^{r-1}$. In computing the weights, we treat G_i as a parameter. In the case of independent kernels, Eq. 17 becomes

$$\tilde{E}_{p'}^r \mathcal{W}(s, p'; G_i^r, \theta_c^{*r}) = \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_i^{*l}, \theta_c^{*l}) \times \frac{K_h(\theta_c^{*l}, \theta_c^{*r}) K_h(G_i^{*l}, G_i^r)}{\sum_{k=r-N}^{r-1} K_h(\theta_c^{*k}, \theta_c^{*r}) K_h(G_i^{*k}, G_i^r)}.$$

We repeat the same step and obtain the pseudo-likelihood conditional on θ_c^{r-1} and $\{G_i^r\}_{i=1}^I$, $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{r-1})$.
Then, we determine whether or not to accept θ^{*r}. The acceptance probability, λ, is given by

$$\begin{aligned}\lambda &= \min\left\{\frac{\pi(\theta_c^{*r}) \tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r}) q(\theta_c^{r-1}|\theta_c^{*r})}{\pi(\theta_c^{r-1}) \tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{r-1}) q(\theta_c^{*r}|\theta_c^{r-1})}, 1\right\} \\ &= \min\left\{\frac{\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{*r}) q(\theta_c^{r-1}|\theta_c^{*r})}{\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \{G_i^r\}_{i=1}^I, \theta_c^{r-1}) q(\theta_c^{*r}|\theta_c^{r-1})}, 1\right\},\end{aligned}$$

where the second equality follows from the flat priors assumption. If accept, set $\theta_c^r = \theta_c^{*r}$; otherwise, set $\theta_c^r = \theta_c^{r-1}$.

6. Computation of the pseudo-$E_\epsilon max$ function, $\{\tilde{\mathcal{W}}^r(., \tilde{p}^r; G_i^{*r}, \theta_c^{*r})\}_{i=1}^I$.
   (a) Make one draw of prices, $\tilde{p}^r$, from the price distribution. Start with consumer $i = 1$.
   (b) Compute the pseudo-expected value function at $(G_i^{*r}, \theta_c^{*r})$. For all s,

$$\tilde{E}_{p'}^r \mathcal{W}(s, p'; G_i^{*r}, \theta_c^{*r}) = \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_i^{*l}, \theta_c^{*r}) \times \frac{K_h(\theta_c^{*l}, \theta_c^{*r}) K_h(G_i^{*l}, G_i^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta_c^{*k}, \theta_c^{*r}) K_h(G_i^{*k}, G_i^{*r})}.^{23}$$

   We should emphasize that there is a subtle difference between this step and step 5(b) above, which evaluates $\tilde{E}_{p'}^r \mathcal{W}$ at G_i^r instead of G_i^{*r}.
   (c) Compute the pseudo-alternative specific value function, $\tilde{W}_j^r(., \tilde{p}^r; G_i^{*r}, \theta_c^{*r})$ using $\tilde{E}_{p'}^r \mathcal{W}$ obtained from (b).
   (d) Then compute the pseudo-$E_\epsilon max$ function:

$$\tilde{\mathcal{W}}^r(., \tilde{p}^r; G_i^{*r}, \theta_c^{*r}) = \ln\left[\sum_{j=0}^{2} \exp(\tilde{W}_j^r(., \tilde{p}^r; G_i^{*r}, \theta_c^{*r}))\right].$$

[23]Note that both the common and individual-specific parts of the weights have already been computed separately in steps 4 and 5, and can thus be re-used here.

(e) Repeat (b)–(d) for all i.
(f) Update H^{r+1}.

7. Go to iteration $r + 1$ (i.e., start step 1).

Here we make four practical suggestions about how to implement the above procedure. First, in step 5(b), one can re-use the individual pseudo-likelihoods already computed in step 4(b) to form the joint pseudo-likelihood conditional on θ_c^{r-1} and $\{G_i^r\}_{i=1}^I$. Second, when implementing step 5, it could be more efficient to separate the parameters by blocks if the acceptance rate is low. The trade-off is that when implementing this step by blocks, we also increase the number of expected future value approximation calculations and likelihood evaluations. Third, the kernel weights based on θ_c are common across all consumers in steps 4 and 5, and can therefore be pre-computed prior to these two steps. This can save some computational time by avoiding double-calculation.

Fourth, the above procedure is much more memory intensive and computationally demanding than the homogeneous case, because we need to store and compute $I \times N$ instead of N past pseudo-$E_\epsilon max$ functions. If computer memory is a constraint faced by a researcher, an alternative procedure is to randomly pick one consumer's pseudo-$E_\epsilon max$ function to store for each iteration l. When approximating the expected value function for each consumer, one can then treat all the past pseudo-$E_\epsilon max$ functions stored as a common pool to form the kernel approximation. In Appendix B, we explain how to implement this procedure.

Finally, it is worth emphasizing that when estimating a model with unobserved heterogeneity, a key difference from the homogeneous model is that one needs to draw individual-specific parameters. When the number of individuals is large, this part of the MCMC algorithm can be slow even for static choice models (e.g., Rossi et al. 2005). When estimating a DDP model, this part of the MCMC algorithm is even more computationally intensive than estimating a static choice model, because the expected value functions need to be computed individual-by-individual. This feature suggests that the IJC method may be particularly useful in estimating models with unobserved heterogeneity. Suppose that we can save one second by using the IJC method to approximate one expected value function in the inner loop. If there are 1,000 individuals in the sample, we can then approximately save 1,000 s per MCMC outer loop iteration. In our Monte Carlo exercises, we will illustrate the potential time savings that one can achieve by using the IJC method.

4.4 Extension to serially correlated prices

So far we assume prices are distributed *i.i.d.* over time. It is straightforward to extend the IJC algorithm to incorporate serially correlated prices. To illustrate how it works, we consider the homogeneous model here.

Suppose that prices set by the two supermarket chains follow a first-order Markov process: $f(p'|p; \theta_p)$, where θ_p is the vector of parameters for the price process. In this setting, the expected value functions in Eqs. 15 and 16 are conditional on the current price p, $E_{p'}[\mathcal{W}(., p'; \theta)|p]$. To handle this situation, for each iteration r, we make one draw of prices, $\tilde{p}^r = (\tilde{p}_1^r, \tilde{p}_2^r)$, from a uniform distribution on $[\underline{p}, \bar{p}]^2$ where $\underline{p}$ and $\bar{p}$ are the lowest and highest observed prices, respectively. Then, we compute and store the pseudo-$E_\epsilon max$ function at $\tilde{p}^r$, $\tilde{\mathcal{W}}^r(., \tilde{p}^r; \theta^{*r})$, and also store $\tilde{p}^r$. Thus, H^r in step 1 of Section 4.3.1 needs to be modified as

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

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

$$\tilde{E}_{p'}^r[\mathcal{W}(s', p'; \theta^{*r})|p] = \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s', \tilde{p}^l; \theta^{*l}) \frac{K_h(\theta^{*l}, \theta^{*r}) f(\tilde{p}^l|p; \theta_p)}{\sum_{k=r-N}^{r-1} K_h(\theta^{*k}, \theta^{*r}) f(\tilde{p}^k|p; \theta_p)}. \tag{18}$$

Compared with Eq. 17, the main difference is that in Eq. 18, the transition density of prices is also used in creating the weights so as to integrate out future prices.

4.5 Choice of kernel's bandwidth and N

The IJC method relies on classical non-parametric methods to approximate the expected value function. One practical problem of nonparametric regression analysis is that the data becomes increasingly sparse as the dimensionality of the explanatory variables increases (note that the number of parameters of the DDC model in IJC corresponds to the number of explanatory variables in the traditional non-parametric regression). 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 a 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.[24]

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-$E_\epsilon max$ functions (or pseudo-value functions in the general case) generated by the algorithm (by increasing N). In practice, it

[24]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 linearly with the number of values for each state variable.

is possible that N may need to increase with the number of parameters in the model to ensure that the IJC algorithm performs well. As a result, it would also take more time to compute one iteration if the model becomes more complicated.

The discussion above suggests that the convergence rate is typically inversely related to the number of dimensions. But the situation that we face here 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 a model is well-identified and the data has 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 expected pseudo-$E_\epsilon max$ functions would be evaluated at parameter vectors that are concentrated in a small neighborhood in the parameter space. We expect that this should alleviate the curse of dimensionality problem.

It is worth discussing the impact of N on the estimation results. If we increase N, older past pseudo-$E_\epsilon max$ functions will be used in the approximation. This may result in slow improvements in the approximation, and may slow down the MCMC convergence rate. If we decrease N, more recent and accurate pseudo-$E_\epsilon max$ functions will be used in the approximation. However, by decreasing N, the variance of the pseudo-expected value functions may 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 a small N 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 weigh the past N expected pseudo-$E_\epsilon max$ functions differently so that the more recent expected pseudo-$E_\epsilon max$ functions receive higher weights (because they should be more accurate approximations).

An obvious question that would likely come to researchers' mind is: How do we choose N and h (i.e., the bandwidth for the kernel)? We believe that any suggested guidelines should ensure that the pseudo-$E_\epsilon max$ function gives us a good approximation for the true $E_\epsilon max$ function. We suggest that researchers check the distance between the pseudo-$E_\epsilon max$ function and the exact $E_\epsilon max$ function during the estimation, and adjust N and h within the iterative process. For instance, researchers can store a large set of past pseudo-$E_\epsilon max$ 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). For every, say 1,000 iterations, they can compute the means of the MCMC draws, $\bar{\tilde{\theta}}$, and solve for the exact $E_\epsilon max$ function at $\bar{\tilde{\theta}}$ numerically. Then they can compare the distance between the pseudo-$E_\epsilon max$ function and the exact $E_\epsilon max$ function at $\bar{\tilde{\theta}}$. If the distance is larger than what the researcher would accept, N' should be increased. Researchers can then use this new larger set of past pseudo-$E_\epsilon max$ functions to compute summary statistics and apply a 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-$E_\epsilon max$ function is that it requires more memory. 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 become less of a constraint in the near future. This suggestion would require us to solve for the DDP model exactly once in every 1,000 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-$E_\epsilon max$ function and the exact $E_\epsilon max$ 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.

5 Estimation results

To illustrate the performance of the IJC algorithm and investigate some of its properties, we conduct two sets of 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-$E_\epsilon max$ functions when approximating the expected value 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. For all experiments, the following parameters are fixed and not estimated: $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\bar{p} = 1.0$, and $\sigma_p = 0.3$. In the first set of experiments, we check whether IJC is able to recover the true parameter values of the model presented here.

5.1 Ability of recovering true parameter values

We first estimate a version of the model *without unobserved heterogeneity*. When simulating the data, we set $\sigma_{G_1} = \sigma_{G_2} = \alpha_1 = \alpha_2 = 0.0$, $G_1 = 1.0$, $G_2 = 5.0$, $\gamma = -1.0$, 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 $\beta < 1$ during the estimation, we transform it as $\beta = \frac{1}{1+\exp(\phi)}$ and estimate ϕ instead. For all parameters, we use the flat prior (i.e., $\pi(\theta) = 1, \forall\theta$). In addition, we use a normal random-walk proposal distribution (i.e., $\theta^{*r} \sim N(\theta^{r-1}, \sigma^2)$). This implies that the acceptance probability stated in step 3 of Section 4.3.1 becomes $\min\left\{\frac{\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{*r})}{\tilde{L}^r(\mathsf{b}|\mathsf{s},\mathsf{p};\theta^{r-1})}, 1\right\}$ because the proposal distribution is symmetric. Table 3 summarizes the estimation results, and Fig. 4 plots the MCMC draws of parameters for the case of $\beta = 0.8$. The posterior means and standard deviations

Table 3 Estimation results: homogeneous model

Parameter	TRUE	$\beta = 0.6$		$\beta = 0.8$	
		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 $j = 1, 2$

Tuning parameters: $N = 1{,}000$ (number of past pseudo-$E_\epsilon max$ functions used for expected value function approximations), $h = 0.01$ (bandwidth)

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.

Now we estimate a version of the model *with unobserved heterogeneity*. For simplicity, we only allow for consumer heterogeneity in G_2 (i.e., we fix $\sigma_{G_1} = 0$). 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. As before, 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 $s_0 = 1$, $\nu_0 \to 1$). In step 4(b) of Section 4.3.2, we use $N(G_j^r, (\sigma_{G_j}^r)^2)$ as the proposal distribution for generating G_{ij}^{*r}. Given this proposal distribution, the probability of accepting G_{ij}^{*r} becomes $\lambda = \min\left\{\frac{\tilde{L}_i^r(\mathrm{b}_i|\mathrm{s}_i,\mathrm{p}_i;G_i^{*r},\theta_c^{r-1})}{\tilde{L}_i^r(\mathrm{b}_i|\mathrm{s}_i,\mathrm{p}_i;G_i^{r-1},\theta_c^{r-1})}, 1\right\}$. In each iteration, we also implement the procedure in Appendix B by randomly selecting one consumer's pseudo-$E_\epsilon max$ function to compute and store, and use a common pool of past pseudo-$E_\epsilon max$ functions to approximate the expected value functions for all consumers. Table 4 shows the estimation results, and Fig. 5 plots the simulated draws of parameters for $\beta = 0.8$. Again, the IJC algorithm 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.

Finally, note that when $\beta = 0.98$, the true parameter values are recovered less precisely, in particular, for α_j and G_j. This is due to an identification problem (Ching et al. 2012): When β is very close to 1, changing G_j would simply shift the choice probabilities almost equally across s, similar to changing α_j.

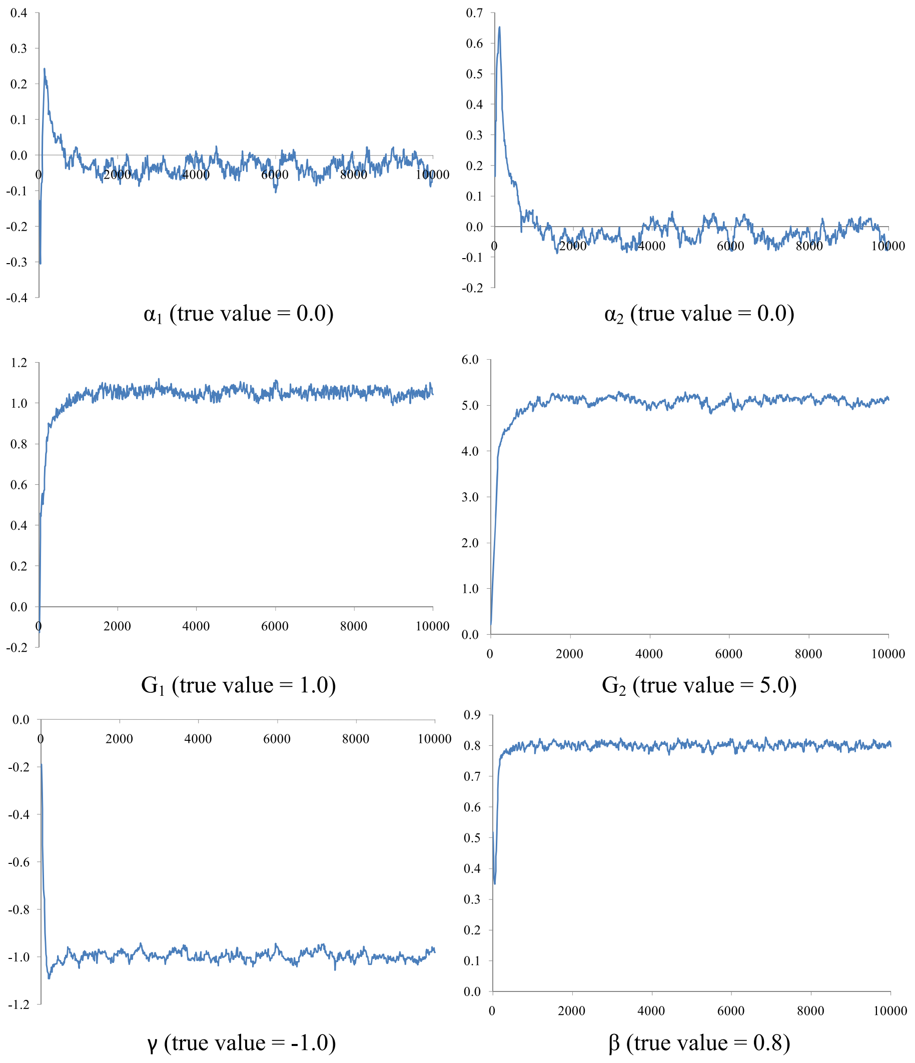

Fig. 4 MCMC plots: homogeneous model with $\beta = 0.8$

5.2 Potential reduction in computation time

In the second set of experiments, we compare the computational time between the IJC and conventional Bayesian approaches. To learn more about the potential gain of IJC in terms of computational time, we compute the time per iteration and compare the IJC algorithm with the full-solution based Bayesian MCMC algorithm for both homogeneous model and heterogeneous model. In

Table 4 Estimation results: heterogeneous model

Parameter	TRUE	$\beta = 0.6$		$\beta = 0.8$	
		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: $\bar{S}_1 = 2$, $\bar{S}_2 = 4$, $\bar{p} = 1.0$, $\sigma_p = 0.3$, $\sigma_{G_1} = 0$

Tuning parameters: $N = 1{,}000$ (number of past pseudo-$E_\epsilon max$ functions used for expected value function approximations), $h = 0.01$ (bandwidth)

the full-solution based Bayesian algorithm, we set $M = 100$ for $\{\tilde{p}^m\}_{m=1}^M$, that is, we use 100 simulated draws of prices to integrate out future prices. For each model, we study three cases: $\beta = 0.6, 0.8$ and 0.98. Table 5 summarizes the average computation time per iteration (in seconds). The estimation is done with a C program compiled by the gcc compiler, and run in a linux workstation with Intel Core 2 Duo E4400 2GHz processor (single-thread).

In the homogeneous model, the computation for the full-solution based Bayesian algorithm is faster for $\beta = 0.6$ and 0.8. This is because: (i) when β is small, solving the exact expected value function is not that costly compared with computing the weighted average of 1,000 past pseudo-$E_\epsilon max$ functions; (ii) full-solution based Bayesian approach does not need to compute the pseudo-likelihood conditional on previously accepted parameter vector (step 3 in the homogeneous case, and step 4(b) in the heterogeneous case).[25] However, when $\beta = 0.98$, the IJC algorithm is 40% faster than the full-solution algorithm. This is because in the full-solution based Bayesian algorithm, the number of iterations required for convergence in a contraction mapping increases with β (i.e., the modulus), and hence the computation time of the inner loop will generally increase with β. However, the computation time of the inner loop will not be influenced by the value of β in the IJC algorithm.

In the heterogeneous model, we can see the advantage of the IJC algorithm much more clearly. 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 3,000% 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

[25] In this exercise, we computed the pseudo-likelihood conditional on previously accepted parameter vector every time a candidate parameter vector was rejected.

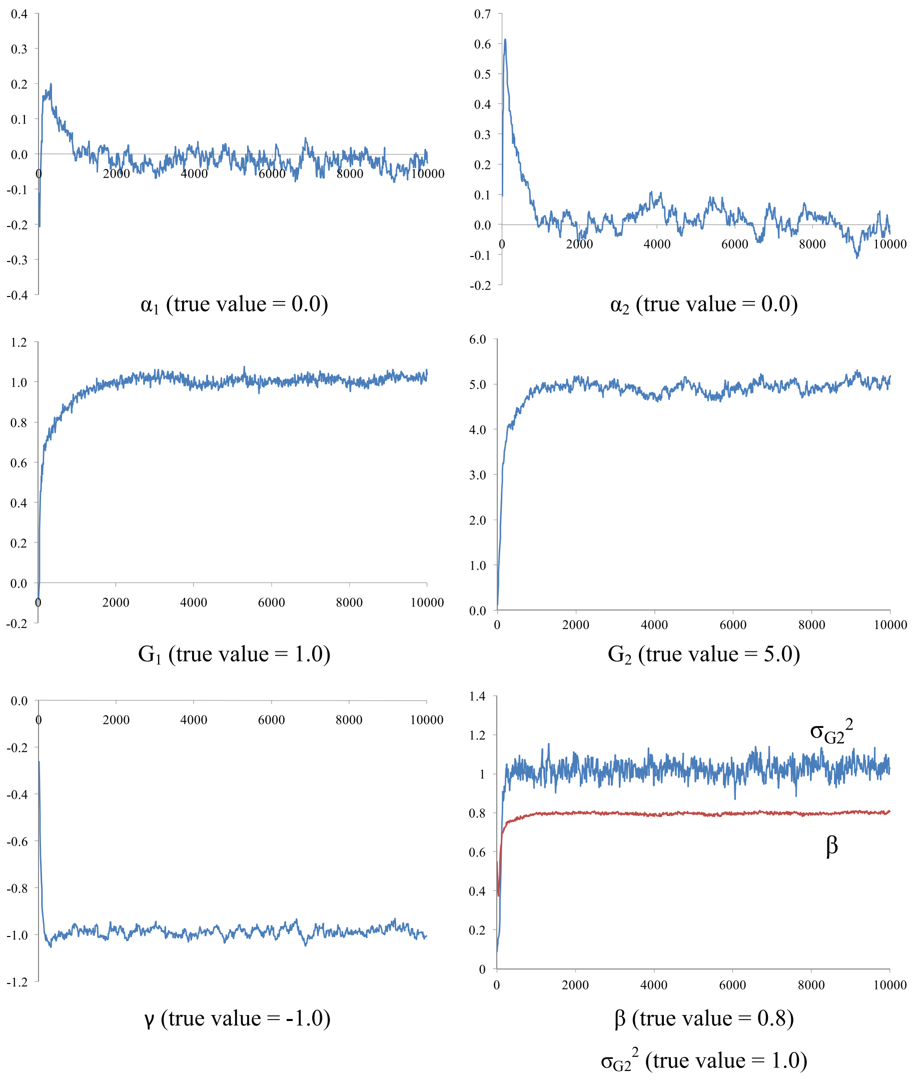

Fig. 5 MCMC plots: Heterogeneous Model with $\beta = 0.8$

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 is approximately equal to the *time to solve the expected value function once* multiplied by one thousand. For the heterogeneous model, with $\beta = 0.98$, it would take about 70 days ($\simeq 613 \times 10{,}000$ s) to run the full-solution based Bayesian MCMC algorithm for 10,000 iterations. Using the IJC algorithm, it would take less than 2.5 days ($\simeq 18.4 \times 10{,}000$ s) to obtain 10,000 iterations.

Table 5 Computation time per MCMC iteration (in seconds)

Algorithm	Homogeneous model			Heterogeneous model		
	$\beta = 0.6$	$\beta = 0.8$	$\beta = 0.98$	$\beta = 0.6$	$\beta = 0.8$	$\beta = 0.98$
Full solution based Bayesian	0.782	0.807	1.410	31.526	65.380	613.26
IJC with $N = 1{,}000$	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 ($\bar{S}_1 = 2$, $\bar{S}_2 = 4$)

Parameters: Homogeneous model:($\alpha_1, \alpha_2, G_1, G_2, \gamma, \beta$). We drew each parameter separately using the Metropolis-Hastings within Gibbs. Heterogeneous model: ($\alpha_1, \alpha_2, G_1, G_2, \sigma_{G_2}, \gamma, \beta$). We drew each parameter except for G_2 and σ_{G_2} separately using the Metropolis-Hastings within Gibbs

6 Comparison with other approximation approaches

In this section, we will compare the IJC algorithm with two other estimation approaches by Keane and Wolpin (1994) and Ackerberg (2009), which also rely on approximating the likelihood.

6.1 Keane and Wolpin (1994)

So far we have focused on estimating infinite horizon stationary DDP models. The estimation issues for finite horizon non-stationary DDP models are slightly different.[26] When solving a finite horizon non-stationary DDP model, the common approach is to use backward induction and start solving the model from the terminal period, T. Unlike the infinite horizon DDP models, which rely on the successive approximation algorithm to obtain the value function, the computational burden of solving finite horizon DDP models does not depend on the discount factor, β. Instead, it solely depends on the size of the state space. As shown above, the IJC algorithm can be viewed as an extension of the successive approximation algorithm. Therefore, it may first seem that the IJC algorithm would not be suitable for estimating a finite horizon DDP model. However, Ishihara (2011) points out that when *some* of the state variables are *continuous* and evolve *stochastically*, it is still possible to extend the IJC algorithm to reduce the computational burden of estimating such a model.

Another method to reduce the computational burden for this type of model is by Keane and Wolpin (1994). Their method is more general in the sense that it applies even though all the state variables are discrete. Moreover, it is designed to handle problems with large state spaces. In their approach, the

[26]Examples of finite horizon non-stationary dynamic programming models include Ching (2010), Diermeier et al. (2005), Keane and Wolpin (1997) and Yang and Ching (2010). It is typical to use this approach when modeling agents' decisions during their life-cycles.

expected value functions are evaluated at a subset of state points and some methods of interpolation are used to evaluate the expected value functions at other values of the state space. Keane and Wolpin (1994) provide Monte Carlo evidence that suggests their approximation would converge to the true solution as the subset of the state points that are chosen increases. This method has proven to be very effective in reducing the computational time for finite horizon DDP models with a large state space (e.g., Erdem and Keane 1996; Keane and Wolpin 1997; Ackerberg 2003; Crawford and Shum 2005). One fruitful area for future research is to combine this interpolation approach with the IJC algorithm. This could address the memory constraint limitation for the IJC algorithm.

6.2 Ackerberg (2009)

Recently, Ackerberg (2009) proposed an alternative estimation approach that makes use of importance sampling and change of variables techniques. To use this approach, one would simulate a set of parameter vectors, $\{\tilde{\theta}^m\}_{m=1}^M$, solve the model and obtain the sub-likelihood at each $\tilde{\theta}^m$ upfront. The likelihood is a weighted average of these sub-likelihoods, where the weights are partly determined by the importance sampling density chosen. When searching over the parameter space to maximize the likelihood, one only needs to change the weights associated with each sub-likelihood, and does not have to solve the model at different simulated parameter vectors again in each outer loop iteration. As a result, Ackerberg's approach also has the potential to significantly reduce the computational burden of estimating DDP models. This method has been applied to several DDP problems, e.g., Hartmann (2006) and Pantano (2008).

Ackerberg's approach is more general than IJC in the sense that it can be applied to a larger class of DDP models, including finite horizon non-stationary DDP models with discrete state space. However, often times, to apply his method, one needs to allow all parameters of the model to have continuous random effects, and this could lead to identification problems for some of the parameters. Moreover, a poor choice of importance sampling density function could lead to very large simulation errors in this approach. Ackerberg (2009) discusses some practical ways to address this issue.

It should also be noted that the IJC method can be used in both Bayesian and classical estimation, while Ackerberg's method is less suitable for the Bayesian approach. This is because in the Bayesian approach, one does not have to marginalize the unobserved heterogeneity. Rather, it is more straightforward to treat the random coefficient as a set of individual-specific parameters in the Hierarchical Bayes approach. For models with complicated likelihoods (i.e., with multiple modes), the MCMC based IJC approach might lead to more stable estimation results because, in practice, simulating draws from posterior distributions using MCMC appears to be easier than searching for the global maximum in such situations.

7 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 (2011) has successfully applied this method to estimate a much more detailed consumer learning model. Roos et al. (2011) have applied it to estimate a hyper-media search model. The IJC method allows them to incorporate 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 micro panel data. Ching et al. (2009) have also successfully estimated a learning and forgetting model where consumers are forward-looking.

It should also be noted that there are many kernels that one could use in forming a non-parametric approximation for the expected value functions. IJC discuss their method in terms of the Gaussian kernel. Norets (2009) extends IJC's method by using the nearest neighbor kernel instead of Gaussian kernel, 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-$E_\epsilon max$ 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.

The IJC approach is most useful when applying to stationary DDP models. In principle, the IJC method should also be applicable to any structural models which can be solved using contraction mapping arguments. For instance, Berry et al. (1995) propose to use the method of successive approximation to back out the unobserved product characteristics in their nested fixed point GMM approach when estimating static demand models using product level data; in some game-theoretic models (e.g., bargaining models) which have a unique equilibrium, the method of successive approximation is also a common way to numerically solve for the equilibrium. The IJC method can be potentially applicable in these situations as well.

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 DDP models in the near future.

Acknowledgements We thank Martin Burda, Monica Meireles, Matthew Osborne, Peter Rossi, Andrei Strijnev, K. Sudhir, S. Siddarth and two anonymous referees for their helpful comments. We also thank the participants of the UCLA Marketing Camp, SBIES conference, Marketing Science Conference, Marketing Dynamics Conference, UTD-FORMS Conference, Canadian Economic Association Meeting, Econometric Society Meeting and Ph.D. seminars at OSU's Fisher College of Business, Yale School of Management, University of Groningen, University of

Zurich and University of Southern California for their useful feedback. Hyunwoo Lim provided excellent research assistance. All remaining errors are ours. Andrew Ching and Susumu Imai acknowledge the financial support from SSHRC.

Appendix A

In this appendix, we discuss some techniques that one can use in practice to reduce the computational burden further. While we will use the model without unobserved heterogeneity for illustration purpose, the same ideas apply to the model with unobserved heterogeneity.

A.1 Integration of iid price shocks

In the base model specification of the store choice model with reward programs, we assume that prices are iid normal random variable. When implementing the IJC algorithm, we propose to make one draw of price vector, $\tilde{p}^r$, and store $\tilde{\mathcal{W}}^r(s, \tilde{p}^r; \theta^{*r})$ in each iteration. Alternatively, we may draw a number of price vector in each iteration, $\{\tilde{p}^m\}_{m=1}^M$, evaluate $\bar{E}_{p'}\tilde{\mathcal{W}}^r(s, p'; \theta^r)$ using

$$\bar{E}_{p'}\tilde{\mathcal{W}}^r(s, p'; \theta^{*r}) = \frac{1}{M}\sum_{m=1}^{M}\tilde{\mathcal{W}}^r(s, \tilde{p}^m; \theta^{*r}), \tag{19}$$

and store $\bar{E}_{p'}\tilde{\mathcal{W}}^r(s, p'; \theta^{*r})$ instead of $\tilde{\mathcal{W}}^r(s, \tilde{p}^r; \theta^{*r})$. The expected value function can then be approximated as follows (correspond to step 3 in Section 4.3.1).

$$\tilde{E}^r_{p'}\mathcal{W}(s, p'; \theta^{*r}) = \sum_{l=r-N}^{r-1}\bar{E}_{p'}\tilde{\mathcal{W}}^l(s, p'; \theta^{*l})\frac{K_h(\theta^{*l}, \theta^{*r})}{\sum_{k=r-N}^{r-1}K_h(\theta^{*k}, \theta^{*r})}.$$

In this alternative approach, we integrate out price first, before using the kernel regression to obtain the pseudo expected value function $\tilde{E}^r_{p'}\mathcal{W}(s, p'; \theta^{*r})$. So this approach should allow us to achieve the same level of precision by using a smaller N. One potential advantage is that it saves us some memory when computing the weighted average. The additional cost is that we need to compute $\bar{E}_{p'}\tilde{\mathcal{W}}^r$ in each MCMC iteration. In terms of computational time, we find that these two approaches are roughly the same in our example.

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 $\bar{E}_{p'}\tilde{\mathcal{W}}^r(s, p'; \theta^{*r})$, provided that there are a sufficient number of observations for each s. The advantage of using this approach is that the pseudo-$E_\epsilon max$ functions of the observed prices, $\tilde{W}^r_j(s, \mathsf{p}; \theta^{*r})$, are by-products of the likelihood function computation. So we can skip step 4(a) and (b) in Section 4.3.1.

A.2 Computation of $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$

In Section 4.3.1, we propose to compute the pseudo-likelihood at previously accepted parameter vector, $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$, in each iteration. This is mainly because in IJC, the set of past pseudo-$E_\epsilon max$ functions is updated in each iteration, and thus the pseudo-likelihood computed in the previous iteration, $\tilde{L}^{r-1}(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$, is different from $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$. However, in practice, the computation of pseudo-likelihood is the most time-consuming part in the algorithm. Moreover, the set of past pseudo-$E_\epsilon max$ functions is updated only by one element in each iteration. Thus, we propose the following procedure, which avoids computing $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ in every iteration.

Suppose that we are in step 3 of iteration r (Section 4.3.1). If we have accepted the candidate parameter value in iteration $r-1$ (i.e., $\theta^{r-1} = \theta^{*(r-1)}$), then use $\tilde{L}^{r-1}(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*(r-1)})$ as a proxy for $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$. Note that the calculations of $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ and $\tilde{L}^{r-1}(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*(r-1)})$ only differ in one past pseudo-$E_\epsilon max$ function, and $\tilde{L}^{r-1}(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{*(r-1)})$ has already been computed in iteration $r-1$. If we have rejected the candidate parameter vector (i.e., $\theta^{r-1} = \theta^{r-2}$), then we could use $\tilde{L}^{r-1}(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-2})$ as a proxy for $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$, and only compute $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$ once every several successive rejections. This procedure avoids using the pseudo-likelihood that is based on an old set of past pseudo-$E_\epsilon max$ functions as a proxy for $\tilde{L}^r(\mathsf{b}|\mathsf{s}, \mathsf{p}; \theta^{r-1})$. According to our experience, one can obtain a fairly decent reduction in computational time when using this approach.

Appendix B

In this appendix, we explain an alternative way to implement IJC when estimating the model with unobserved heterogeneity. The main goal of this alternative approach is to reduce the memory requirement and computational burden further. Instead of storing $\{\theta_c^{*l}, \{G_i^{*l}, \tilde{\mathcal{W}}^l(., p^l; G_i^{*l}, \theta_c^{*l})\}_{i=1}^{I}\}_{l=r-N}^{r-1}$, one can store $\{\theta_c^{*l}, G_{i'}^{*l}, \tilde{\mathcal{W}}^l(., p^l; G_{i'}^{*l}, \theta_c^{*l})\}_{l=r-N}^{r-1}$, where $i' = r - I * int(\frac{r-1}{I})$; $int(.)$ is an integer function that converts any real number to an integer by discarding its value after the decimal place. i' is simply one way to "randomly" select a consumer's pseudo-$E_\epsilon max$ function to be stored in each iteration. When approximating the expected value function in, say step 4(b) in Section 4.3.2, we can then set

$$\tilde{E}_{p'}^r \mathcal{W}(s, p'; G_i^{*r}, \theta_c^{r-1}) = \sum_{l=r-N}^{r-1} \tilde{\mathcal{W}}^l(s, \tilde{p}^l; G_{i'}^{*l}, \theta_c^{*l}) \times \frac{K_h(\theta_c^{*l}, \theta_c^{r-1}) K_h(G_{i'}^{*l}, G_i^{*r})}{\sum_{k=r-N}^{r-1} K_h(\theta_c^{*k}, \theta_c^{r-1}) K_h(G_{i'}^{*k}, G_i^{*r})}.$$

Note that we are using the same set of past pseudo-$E_\epsilon max$ functions for all consumers here. If there is a large number of consumers in the sample, this approach, which is also independently adopted by Osborne (2011), can dramatically reduce the memory requirement and computational burden for implementing IJC.

This approach works because $G^{*l}_{i'}$ is a random realization from a distribution that covers the support of the parameter space. This is one important requirement that ensures the pseudo-$E_\epsilon max$ functions converge to the true ones in the proof of IJC.

References

Ackerberg, D. A. (2003). Advertising, learning, and consumer choice in experience good markets: An empirical examination. *International Economic Review, 44*(3), 1007–1040.

Ackerberg, D. A. (2009). A new use of importance sampling to reduce computational burden in simulation estimation. *Quantitative Marketing and Economics, 7*(4), 343–376.

Aguirregabiria, V., & Mira, P. (2002). Swapping the nested fixed point algorithm: A class of estimators for discrete Markov decision models. *Econometrica, 70*(4), 1519–1543.

Albert, J. H., & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association, 88*, 669–679.

Allenby, G. M. (1994). An introduction to hierarchical Bayesian modeling. Tutorial Notes, Advanced Research Techniques Forum, American Marketing Association.

Allenby, G. M., & Lenk, P. J. (1994). Modeling household purchase behavior with logistic normal regression. *Journal of the American Statistical Association, 89*, 1218–1231.

Allenby, G. M., & Rossi, P. E. (2006). Hierarchical Bayes models: A practitioner's guide. In R. Grover, & M. Vriens (Eds.), *The handbook of marketing research*. Newbury Park: Sage Publications.

Berry, S. T., Levinshohn, J., & Pakes, A. (1995). Automobile prices in market equilibrium. *Econometrica, 63*(4), 841–890.

Brown, M., & Flinn, C. J. (2011). Family law effects on divorce, fertility and child investment. 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.

Ching, A. T. (2010). A dynamic oligopoly structural model for the prescription drug market after patent expiration. *International Economic Review, 51*(4), 1175–1207.

Ching, A. T., Imai, S., Ishihara, M., & Jain, N. (2009). A dynamic model of consumer learning and forgetting. Work-in-progress, Rotman School of Management, University of Toronto.

Ching, A. T., Imai, S., Ishihara, M., & Jain, N. (2012). Identification of dynamic models of rewards program. Working paper, Rotman School of Management, University of Toronto.

Crawford, G. S., & Shum, M. (2005). Uncertainty and learning in pharmaceutical demand. *Econometrica, 73*(4), 1137–1174.

Diermeier, D., Keane, M. P., & Merlo, A. M. (2005). A political economy model of congressional careers. *American Economic Review, 95*, 347–373.

Erdem, T., Imai, S., & Keane, M. P. (2003). Brand and quality choice dynamics under price uncertainty. *Quantitative Marketing and Economics, 1*(1), 5–64.

Erdem, T., & Keane, M. P. (1996). Decision making under uncertainty: Capturing dynamic brand choice processes in turbulent consumer goods markets. *Marketing Science, 15*(1), 1–20.

Geweke, J., Houser, D., & Keane, M. P. (2001). Simulation based inference for dynamic multinomial choice models. In B. H. Baltagi (Ed.), *A companion to theoretical econometrics* (pp. 466–493). London: Blackwell.

Geweke, J. F., & Keane, M. P. (2000). Bayesian inference for dynamic discrete choice models without the need for dynamic programming. In R. Mariano, T. Schuermann, & M. J. Weeks (Eds.), *Simulation based inference and econometrics: Methods and applications*. Cambridge: Cambridge University Press.

Gönül, F., & Srinivasan, K. (1996). Estimating the impact of consumer expectations of coupons on purchase behavior: A dynamic structural model. *Marketing Science, 15*(3), 262–279.

Hartmann, W. R. (2006). Intertemporal effects of consumption and their implications for demand elasticity estimates. *Quantitative Marketing and Economics, 4*(4), 325–349.

Hendel, I., & Nevo, A. (2006). Measuring the implications of sales and consumer inventory behavior. *Econometrica, 74*(6), 1637–1673.

Hitsch, G. (2006). An empirical model of optimal dynamic product launch and exit under demand uncertainty. *Marketing Science, 25*(1), 25–50.

Hotz, J. V., & Miller, R. (1993). Conditional choice probabilities and the estimation of dynamic models. *Review of Economic Studies, 60*(3), 497–529.

Imai, S., Jain, N., & Ching, A. (2009a). Bayesian estimation of dynamic discrete choice models. *Econometrica, 77*(6), 1865–1899.

Imai, S., Jain, N., & Ching, A. (2009b). Supplement to 'Bayesian estimation of dynamic discrete choice models'. *Econometrica (Supplementary Material), 77*. http://www.econometricsociety.org/ecta/Supmat/5658_proofs.pdf.

Imai, S., & Krishna, K. (2004). Employment, deterrence and crime in a dynamic model. *International Economic Review, 45*(3), 845–872.

Ishihara, M. (2011). *Dynamic demand for new and used durable goods without physical depreciation*. Ph.D. dissertation, Rotman School of Management, University of Toronto.

Keane, M. P., & Wolpin, K. I. (1994). The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte Carlo evidence. *Review of Economics and Statistics, 76*(4), 648–672.

Keane, M. P., & Wolpin, K. I. (1997). The career decisions of young men. *Journal of Political Economy, 105*, 473–521.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Lancaster, T. (1997). Exact structural inference in optimal job search models. *Journal of Business and Economic Statistics, 15*(2), 165–179.

McCulloch, R., & Rossi, P. E. (1994). An exact likelihood analysis of the multinomial probit model. *Journal of Econometrics, 64*, 207–240.

Norets, A. (2009). Inference in dynamic discrete choice models with serially correlated unobserved state variables. *Econometrica, 77*(5), 1665–1682.

Norets, A. (2010). Continuity and differentiability of expected value functions in dynamic discrete choice models. *Quantitative Economics, 1*(2), 305–322.

Osborne, M. (2011). Consumer learning, switching costs, and heterogeneity: A structural examination. *Quantitative Marketing and Economics, 9*(1), 25–70.

Pantano, J. (2008). Essays in applied microeconomics. Ph.D. dissertation, UCLA.

Roos, J. M. T., Mela, C. F., & Shachar, R. (2011). Hyper-media search and consumption. Working paper, Fuqua School of Business, Duke University.

Rossi, P. E., & Allenby, G. M. (1999). Marketing models of consumer heterogeneity. *Journal of Econometrics, 89*, 57–78.

Rossi, P. E., Allenby, G. M., & McCulloch, R. (2005). *Bayesian statistics and marketing*. Chichester: Wiley.

Rossi, P. E., McCulloch, R., & Allenby, G. M. (1996). The value of purchase history data in target marketing. *Marketing Science, 15*, 321–340.

Rust, J. (1987). Optimal replacement of gmc bus engines: An empirical model of Harold Zurcher. *Econometrica, 55*(5), 999–1033.

Rust, J. (1988). Maximum likelihood estimation of discrete control processes. *SIAM Journal on Control and Optimization, 26*(5), 1006–1024.

Rust, J. (1997). Using randomization to break the curse of dimensionality. *Econometrica, 65*(3), 487–516.

Santos, M. S., & Rust, J. (2004). Convergence properties of policy iteration. *SIAM Journal on Control and Optimization, 42*(6), 2094–2115.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London: Chapman and Hall.

Song, I., & Chintagunta, P. K. (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, B. (2005). Promotion effect on endogenous consumption. *Marketing Science, 24*(3), 430–443.

Train, K. E. (2003). *Discrete choice methods with simulation*. Cambridge: Cambridge University Press. Available at http://elsa.berkeley.edu/books/choice2.html.

Walsh, B. (2004). Markov Chain Monte Carlo and Gibbs sampling. Lecture Notes for EEB 581, University of Arizona. http://nitro.biosci.arizona.edu/courses/EEB581-2004/handouts/Gibbs.pdf.

Yang, B., & Ching, A. (2010). Dynamics of consumer adoption of financial innovation: The case of ATM cards. Working paper, Rotman School of Management, University of Toronto. Available at SSRN: http://ssrn.com/abstract=1434722.