Estimation of Dynamic Discrete Games Using the Nested Pseudo Likelihood Algorithm: Code and Application

Victor Aguirregabiria

University of Toronto, Department of Economics

15. September 2009

Online at http://mpra.ub.uni-muenchen.de/17329/
Estimation of Dynamic Discrete Games Using the Nested Pseudo Likelihood Algorithm: Code and Application

Victor Aguirregabiria*
University of Toronto

August 20, 2009

Abstract

This document describes program code for the solution and estimation of dynamic discrete games of incomplete information using the Nested Pseudo Likelihood (NPL) method in Aguirregabiria and Mira (2007). The code is illustrated using a dynamic game of store location by retail chains, and actual data from McDonalds and Burger King.

TABLE OF CONTENTS

1. Introduction
2. Empirical Application
   2.1. Model
   2.2 Data
   2.3. NPL Method
3. Main Program (npl_dyngame.prg)
   Part 2: Reading data. Construction of vectors of observed states and decisions
   Part 3: Procedures
   Part 4: Initial Estimates of Conditional Choice Probabilities
   Part 5: NPL Estimation
4. Procedures
   4.1. freqprob
   4.2. miprobit
   4.3. npl_bkmd
5. Output and Empirical Results
Appendix: Gauss code

*Department of Economics, University of Toronto. E-mail: victor.aguirregabiria@utoronto.ca
1 Introduction

This document describes program code for the solution and estimation of dynamic discrete games of incomplete information using the Nested Pseudo Likelihood (NPL) method in Aguirregabiria and Mira (2007). The code is written in GAUSS programming language and it is included in an Appendix and available online at http://individual.utoronto.ca/vaguirre/. Given that the code uses low-level commands in GAUSS, it should be straightforward to translate it to other matrix languages such as Matlab, Fortran 90, R, or C++. I illustrate the use of this software using a dynamic game of store location by retail chains and actual data for McDonalds and Burger King. The example is intentionally simple and it tries to provide a helpful starting point for the user of this code. The list of programs (.prg) and procedures (.src) is the following:

<table>
<thead>
<tr>
<th>Program / Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npl_dyngame.prg</td>
<td>Main program for the NPL estimation of a dynamic game of store location</td>
</tr>
<tr>
<td>npl_bkmd.src</td>
<td>Given an initial vector of choice probabilities (CCPs), it computes an NPL fixed point estimator.</td>
</tr>
<tr>
<td>miprobit.src</td>
<td>Given a vector of choice probabilities (CCPs), it returns the pseudo ML estimator of a probit model.</td>
</tr>
<tr>
<td>freqprob.src</td>
<td>Calculates a frequency estimator of Conditional Choice Probabilities (CCPs)</td>
</tr>
</tbody>
</table>

The main program is npl_dyngame.prg. It includes all the procedures that it calls, such that the user does not have to create any GAUSS library with the procedures called by the main program.\footnote{Alternatively, the user might prefer to remove from the main program all the procedures and place them in a GAUSS library.}

The rest of this document is organized as follows. Section 2 presents the model, data, and the estimation method in the empirical application that we use to illustrate the algorithm and code. Section 3 describes the different parts of the main program. Section 4 goes through the procedures or subroutines called by the main program. Section 5 describes the estimation output and comments the estimates in the empirical application. The code is included in an Appendix.
2 Empirical Application

2.1 Model

Time is discrete and indexed by \( t \in \{1, 2, ..., T\} \), where \( T \) is the time horizon. There are two players in the game, and we use the indexes \( i \in \{1, 2\} \) and \( j \in \{1, 2\} \) to represent a player and his opponent, respectively. Every period, each player makes a binary choice. Within a given period players’ actions are taken simultaneously. Let \( Y_{it} \in \{0, 1\} \) represent the choice of player \( i \) at period \( t \).

Each player makes this decision to maximize its expected and discounted payoff \( E_t(\sum_{s=0}^{T-t} \beta^s_i \Pi_{i,t+s}) \), where \( \beta_i \in (0, 1) \) is player \( i \)'s discount factor and \( \Pi_{it} \) is his payoff at period \( t \). Here we concentrate in Markovian decision models with infinite horizon, \( T = \infty \). The payoff function has the following structure:

\[
\Pi_{it} = z_{it}(Y_{it}, Y_{jt}) \theta_i - Y_{it} \varepsilon_{it} \tag{1}
\]

\( z_{it}(0, 0), z_{it}(0, 1), z_{it}(1, 0), \) and \( z_{it}(1, 1) \) are row vectors of known functions of state variables. \( \theta_i \) is a column vector of structural parameters, and \( \theta \equiv (\theta_1, \theta_2) \) is the vector with both players' parameters. Structural parameters and the vectors \( z_{it}(Y_{it}, Y_{jt}) \) are common knowledge to the two players, up to the action of the other player. The variable \( \varepsilon_{it} \) is private information of firm \( i \) at period \( t \). A player has uncertainty on the current value of his opponent’s \( \varepsilon \), and on future values of both his own and his opponent’s \( \varepsilon \)’s. The vectors \( z_{it}(Y_{it}, Y_{jt}) \) have the following structure:

\[
z_{it}(Y_{it}, Y_{jt}) = z(W_i, X_{it}, X_{jt}, Y_{it}, Y_{jt}) \tag{2}
\]

\( z() \) is a known vector-valued function. \( W_i \) is a vector of time-invariant exogenous characteristics of player \( i \). And \( X_{it} \) is an endogenous 'stock' variable for player \( i \) that evolves over time according to the transition rule \( X_{it+1} = X_{it} + Y_{it} \). The set of possible values for these stock variables is \( \{0, 1, 2, ..., K\} \) where \( K > 1 \) is a natural number that represents the maximum level of the stock. The variables \( \varepsilon_{1t} \) and \( \varepsilon_{2t} \) are independent of \( (W_1, W_2) \), independent of each other, and independently and identically distributed over time. Their distribution functions, \( G_1 \) and \( G_2 \), are absolutely continuous and strictly increasing with respect to the Lebesgue measure on \( \mathbb{R} \).

The model can easily accommodate depreciation (e.g., \( X_{it+1} = (1-\delta)X_{it} + Y_{it} \), with \( 0 < \delta < 1 \)) and endogenous disinvestment (e.g., \( Y_{it} \in \{-1, -1, +1\} \)). However, in the data that we use to
illustrate the algorithm and code, the two retail chains never closed a store during the sample period. Therefore, we have preferred to present here the simple case without depreciation or disinvestment because that is the case in the empirical application.

**EXAMPLE (Capacity Investment in an Oligopoly Industry).** Consider a dynamic game of capacity investment between two firms competing in an oligopoly industry of an homogeneous product. The demand function is \(Q_t = S_t(b_0 - b_1P_t)\), where \(b_0\) and \(b_1\) are parameters, \(Q_t\) represents aggregate output, \(S_t\) is the exogenous market size, and \(P_t\) is the product price. There are 2 firms operating in the industry. Every period \(t\), these firms compete in quantities a la Cournot (static game), and choose whether to invest to increase their capacity (dynamic game). Production costs are linear in the quantity produced, i.e., \(C_{it} = MC_{it}q_{it}\), where \(MC_{it}\) is the marginal cost, and \(q_{it}\) represents output. Marginal cost declines with installed capacity, i.e., \(MC_{it} = c_i - d(X_{it} + Y_{it})\), where \(c_i > 0\) and \(d > 0\) are parameters, \(X_{it}\) is the installed capacity at the beginning of period \(t\), and \(Y_{it} \in \{0, 1\}\) represents capacity investment, that is a binary choice. It is simple to show that the Cournot equilibrium variable profit of firm \(i\) is:

\[
VP_{it} = \frac{S_t}{b_1} \left( \frac{b_0 + MC_{jt} - MC_{it}}{3} \right)^2
\]

\[
= \theta_{0i}^{VP} S_t \{1\{X_{it} + Y_{it} > 0\} + \theta_{1i}^{VP} \{S_t(X_{it} + Y_{it} - X_{jt} - Y_{jt}) + \theta_{2i}^{VP} \} S_t(X_{it} + Y_{it} - X_{jt} - Y_{jt})^2 \}
\]

where \(1\{.\}\) is the indicator function, and \(\theta_{0i}^{VP} > 0\), \(\theta_{1i}^{VP}\), and \(\theta_{2i}^{VP}\) are structural parameters that are known functions of the 'deep' parameters \(b_0\), \(b_1\), \(c_i\), \(c_j\), and \(d\). More specifically, it is simple to verify that \(\theta_{0i}^{VP} \equiv (b_0 + c_j - c_i)^2\), \(\theta_{1i}^{VP} \equiv 2d(b_0 + c_j - c_i)\), and \(\theta_{2i}^{VP} \equiv d^2\). Given a value of the vector of parameters \((\theta_{0i}^{VP}, \theta_{1i}^{VP}, \theta_{2i}^{VP} : i = 1, 2)\), we can (over-)identify the 'deep' structural parameters \(d\), \(b_0\), and \((c_j - c_i)\). Here we concentrate on the identification and estimation of the parameters \((\theta_{0i}^{VP}, \theta_{1i}^{VP}, \theta_{2i}^{VP} : i = 1, 2)\) together with the parameters in fixed costs. The set of possible capacity levels

\(^2\)See Besanko and Doraszelski (2004), or Ryan (2009) for related dynamic games of firm capacity.

\(^3\)We may consider a more flexible model of competition between McDonalds and Burger King. Suppose that these firms have differentiated products. The demand function form firm \(i\) is \(q_i = A_i - b(P_i - P_j)\), where \(A_i\) and \(b\) are structural demand parameters. The specification of the marginal cost function is the same as above. Firms compete in prices a la Nash-Bertrand. It is straightforward to show that variable profit of firm \(i\) in the Nash-Bertrand equilibrium is:

\[
VP_i = \frac{S}{b} \left( \frac{2A_i + A_j + b(MC_i - MC_j)}{3} \right)^2
\]

This expression is useful to interpret the empirical results. In particular, some differences between the parameters.
is \{0, 1, 2, ..., K\} where \(K - 1 > 1\) is a natural number that represents the maximum feasible level of capacity. A firm’s capacity evolves over time according to the transition rule \(X_{it+1} = X_{it} + Y_{it}\).

The firm’s total profit function is:

\[
\Pi_{it} = VP_{it} - \theta_{0i}^{FC} 1\{Y_{it} + X_{it} > 0\} - \theta_{1i}^{FC} (Y_{it} + X_{it}) - \theta_{2i}^{FC} (Y_{it} + X_{it})^2 - Y_{it} \varepsilon_{it}
\]

where \(\theta_{0i}^{FC}, \theta_{1i}^{FC}\) and \(\theta_{2i}^{FC}\) are parameters in the fixed cost function of firm \(i\). The variable \(\varepsilon_{it}\) is a private information shock in the firm’s investment cost, and it is normally distributed. In this example, the vector of structural parameters for firm \(i\) is:

\[
\theta_i \equiv (\theta_{0i}^{VP}, \theta_{1i}^{VP}, \theta_{2i}^{VP}, \theta_{0i}^{FC}, \theta_{1i}^{FC}, \theta_{2i}^{FC})'
\]

and the vector \(Z_{it}(Y_{it}, Y_{jt})\) is:

\[
Z_{it}(Y_{it}, Y_{jt}) \equiv \{ S_t 1\{X_{it} + Y_{it} > 0\}, S_t (X_{it} + Y_{it} - X_{jt} - Y_{jt}), S_t (X_{it} + Y_{it} - X_{jt} - Y_{jt})^2 \\
-1 \{X_{it} + Y_{it} > 0\}, -(X_{it} + Y_{it}), -(X_{it} + Y_{it})^2\}
\]

In our empirical application, we consider the industry of fast-food burger restaurants in UK. The two companies are McDonalds and Burger King who compete in the number of stores. A local market is a district. \(X_{it}\) represents the number of installed stores, and \(Y_{it}\) is the decision to open a new store. During the sample period (1991-1996), these firms did not close any existing store. That is the reason why there is not an exit decision in the model. The model assumes that the decision to open a new store is completely irreversible.

Players’ strategies are the result of a Markov Perfect Equilibrium (MPE). In a MPE, players’ strategies depend only on payoff relevant state variables. In this model, the payoff-relevant information of firm \(i\) at period \(t\) is \((S_t, X_{1t}, X_{2t}, \varepsilon_{it})\). We use \(X_t\) to represent the vector of common knowledge state variables: \(X_t \equiv (S_t, X_{1t}, X_{2t})\). Let \(\mathcal{X}\) be set with all the possible values of \(X_t\). Let \(\sigma \equiv \{\sigma_i(X_t, \varepsilon_{it}) : i = 1, 2\}\) be a set of strategy functions, one for each player. \(\sigma\) is a MPE if, for every player \(i\), the strategy \(\sigma_i\) maximizes the expected value of firm \(i\) at every state \((X_t, \varepsilon_{it})\) and taking as given the opponent’s strategy. It is convenient to represent players’ strategies and MPE in terms of players’ Conditional Choice Probabilities (CCPs). Let \(P_i(X_t)\) represents firm \(i\)’s
probability of increasing its capacity (i.e., of \( Y_{it} = 1 \)) given that the state is \( X_t \). This probability is defined as the integral of the strategy function \( \sigma_i(X_t, \varepsilon_{it}) \) over the distribution of \( \varepsilon_{it} \).

\[
P_i(X_t) \equiv \int 1 \{ \sigma_i(X_t, \varepsilon_{it}) = 1 \} \, dG_i(\varepsilon_{it})
\]  

(7)

where \( 1 \{ \cdot \} \) is the indicator function. We can represent a MPE as a pair of probability functions \( P \equiv \{ P_i(X_t) : i = 1, 2; X_t \in \mathcal{X} \} \) such that the strategy \( P_i \) maximizes the expected value of firm \( i \) at every state \( X_t \) taking as given the opponent’s strategy \( P_j \).

The equilibrium mapping in the space of CCPs is the key component of this class of dynamic games. It summarizes all the relevant structure in the model. The form of this equilibrium mapping depends on the payoff function, the transition rule of the state variables, and the distribution of the private information shocks \( \varepsilon_{it} \). As shown above, in our model the one-period profit of firm \( i \) can be written as \( \Pi_{it} = Z_{it}(Y_{it}, Y_{jt}) \theta_i - Y_{it} \varepsilon_{it} \). Therefore, the one-period expected profit of firm \( i \) is:

\[
\Pi_{it}^P(Y_{it}) = (1 - P_j(X_t)) \, Z_{it}(Y_{it}, 0) \theta_i + P_j(X_t) \, Z_{it}(Y_{it}, 1) \theta_i - Y_{it} \varepsilon_{it}
\]  

(8)

where \( z_{it}^P(Y_{it}) \equiv (S_t, S_t(X_{it} + Y_{it}), (1 - P_j(X_t))S_t(X_{jt} + 0) + P_j(X_t)S_t(X_{jt} + 1), -Y_{it}, -Y_{it}X_{it}) \).

For the sake of illustration, let us consider first the equilibrium mapping for the case when firms are myopic, i.e., \( \beta_1 = \beta_2 = 0 \). The best response function in the space of a player’s action is:

\[
\{ Y_{it} = 1 \} \Leftrightarrow \{ z_{it}^P(1) \theta_i - \varepsilon_{it} \geq z_{it}^P(0) \theta_i \}
\]  

(9)

And in the space of CCPs, firm \( i \)’s best response is:

\[
\Pr(Y_{it} = 1 \mid X_t) = G_i \left( [z_{it}^P(1) - z_{it}^P(0)] \theta_i \right)
\]  

(10)

A MPE in this static/myopic game (i.e., a Bayesian Nash Equilibrium) is a pair of probability functions that solves the system of equations:

\[
P_1(X_t) = G_1 \left( [z_{it}^P(1) - z_{it}^P(0)] \theta_1 \right)
\]  

\[
P_2(X_t) = G_2 \left( [z_{it}^P(1) - z_{it}^P(0)] \theta_2 \right)
\]  

(11)

for every value of \( X_t \). Given our assumptions on the distributions \( G_i \), Brower’s Theorem implies that an equilibrium exits. The model may have multiple equilibria. Note that, for this myopic
or static game, there is a separate system of equations for every value of $X_t$. We could say that for each value of $X_t$ we have a separate equilibrium. As shown below, this is not the case for a dynamic game. In a MPE of a dynamic game, the whole best response probability function of player $i$ depends on the whole probability function of player $j$ at every possible value of $X_t$.

Now, we describe a MPE in a dynamic game where players are forward-looking, i.e., $\beta_i > 0$. Following Aguirregabiria and Mira (2007), a MPE can be described as a vector of CCPs, $P \equiv \{P_i(X_t) : i = 1, 2; X_t \in \mathcal{X}\}$, such that for every firm $i$ and every state $X_t \in \mathcal{X}$ we have that:

$$P_i(X_t) = G_i \left( \left[ \tilde{z}_t^P(1) - \tilde{z}_t^P(0) \right] \theta_i - \left[ \tilde{e}_t^P(1) - \tilde{e}_t^P(0) \right] \right) \quad (12)$$

where $\tilde{z}_t^P(Y_{it})$ is the expected and discounted sum of current and future $z$ vectors $\{z_{it+s}(Y_{it+s}, Y_{jt+s}) : s = 0, 1, 2, \ldots\}$ which may occur along all possible histories originating from the choice of $Y_{it}$ in state $X_t$, if every player behaves according to their CCPs in $P$. More formally,

$$\tilde{z}_t^P(Y_{it}) = z_t^P(Y_{it}) + E \left( \sum_{s=1}^{\infty} \beta^s z_{it+s}^P(Y_{it+s}) \mid X_t, Y_{it} \right) \quad (13)$$

Similarly, $\tilde{e}_t^P(Y_{it})$ is the expected and discounted sum of realizations of $\{\epsilon_{it+s}Y_{it+s} : s = 0, 1, 2, \ldots\}$ originating from the choice of $Y_{it}$ in state $X_t$, when players behave according to their CCPs in $P$:

$$\tilde{e}_t^P(Y_{it}) = E \left( \sum_{s=1}^{\infty} \beta^s \epsilon_{it+s}Y_{it+s} \mid X_t, Y_{it} \right) \quad (14)$$

Now, we describe in detail the exact computation of the values $\tilde{z}_t^P(0), \tilde{z}_t^P(1), \tilde{e}_t^P(0)$, and $\tilde{e}_t^P(1)$, for every possible value of $X_t$ in the space of $\mathcal{X}$. Let $f^P_i(X_{t+1} \mid Y_{it}, X_t)$ be the transition probability of $\{X_t\}$ from the point of view of player $i$ who knows his own current action $Y_{it}$ but ignores the current action of his competitor and only knows that it is a random draw from the probability distribution $P_j(X_t)$. By definition,

$$f^P_i(X_{t+1} \mid Y_{it}, X_t) \equiv 1 \{X_{it+1} = X_{it} + Y_{it}\} P_j(X_t)^{1\{X_{jt+1} = X_{jt+1}\}} (1 - P_j(X_t))^{1\{X_{jt+1} = X_{jt}\}} \quad (15)$$

Define also the value vector $W_{z_i}^P(X_t) \equiv (1 - P_i(X_t))\tilde{z}_t^P(0) + P_i(X_t)\tilde{z}_t^P(1)$, and the scalar value $W_{\epsilon_i}^P(X_t) \equiv (1 - P_i(X_t))\tilde{e}_t^P(0) + P_i(X_t)\tilde{e}_t^P(1)$. It is straightforward to see that, by definition:

$$\tilde{z}_t^P(Y_{it}) = z_t^P(Y_{it}) + \beta \sum_{X_{t+1} \in \mathcal{X}} f^P_i(X_{t+1} \mid Y_{it}, X_t) W_{z_i}^P(X_{t+1}) \quad (16)$$
and
\[ \varepsilon_{it}^P(Y_{it}) \equiv \beta \sum_{X_{it+1} \in X^t} f_i^P(X_{t+1}|Y_{it}, X_t) W_{e_i}^P(X_{t+1}) \]  

(17)

The matrix of values \( W_{Z_i}^P \equiv \{ W_{Z_i}^P(X) : X \in \mathcal{X} \} \) and the vector of values \( W_{e_i}^P \equiv \{ W_{e_i}^P(X) : X \in \mathcal{X} \} \)

are obtained by solving systems of linear equations with dimension \(|\mathcal{X}|\). The solution to these systems of equations has the following closed-form analytical expression:

\[ W_{Z_i}^P = (I - \beta F_X^P)^{-1} [(1 - P_i) * Z_i^P(0) + P_i * Z_i^P(1)] \]  

(18)

and

\[ W_{e_i}^P = (I - \beta F_X^P)^{-1} e_i^P \]  

(19)

\( P_i \) is a \(|\mathcal{X}| \times 1 \) vector with the stacked CCPs of player \( i \) for every possible value of \( X_t \). \( Z_i^P(0) \)

and \( Z_i^P(1) \) are matrices with \(|\mathcal{X}| \) rows and the same number of columns as \( z_{it}^P(Y) \) such that a row of \( Z_i^P(Y) \) is equal to the vector \( z_{it}^P(Y) \) associated with a given value of \( X_t \). * represents the Hadamard or element-by-element product. \( I \) represents the identity matrix with dimension \(|\mathcal{X}|\).

\( F_X^P \) is the transition matrix of \( \{X_t\} \) induced by the vector of CCPs \( P \) such that the elements of this matrix are \((1 - P_i(X_t)) f_i^P(X_{t+1}|0, X_t) + P_i(X_t)) f_i^P(X_{t+1}|1, X_t)\), or what is equivalent,

\[ \prod_{i=1}^{2} P_i(X_t)1\{X_{it+1}=X_{it+1}\} (1 - P_i(X_t))1\{X_{it+1}=X_{it}\} \]

Finally, \( e_i^P \) is a vector that contains the expected values \( E(\varepsilon_{it} Y_{it}|X_t, Y_{it} \text{ optimal}) \) for every value of \( X_t \). These conditional expectations only depend on the probability distribution of \( \varepsilon_{it} \) and on the choice probability \( P_i(X_t) \). For the logit and probit models we have the following closed expressions. When \( \varepsilon_{it} \) is extreme value distributed (logit):

\[ E(\varepsilon_{it} Y_{it}|X_t, Y_{it} \text{ optimal}) = Euler - (1 - P_i(X_t)) \ln (1 - P_i(X_t)) - P_i(X_t) \ln (P_i(X_t)) \]  

(20)

where Euler represents Euler’s constant. And when \( \varepsilon_{it} \) has a standard normal distribution (probit):

\[ E(\varepsilon_{it} Y_{it}|X_t, Y_{it} \text{ optimal}) = \phi \left( \Phi^{-1} (P_i(X_t)) \right) \]  

(21)

where \( \phi(.) \) and \( \Phi^{-1}(.) \) are the PDF and the inverse-CDF of the standard normal.

Equation (12) represents a MPE as a fixed point of a mapping in the space of CCPs. Given our assumptions, Brower’s Theorem guarantees the existence of a MPE. In general, there may be multiple equilibria. This equilibrium mapping is the cornerstone of the NPL estimation method.
2.2 Data

To illustrate the algorithm and code, we estimate a dynamic game of store location by McDonalds (MD) and Burger King (BK) using data for United Kingdom during the period 1991-1995. The dataset comes from the paper Toivanen and Waterson (2005). It is a panel of 422 local markets (districts) and five years with information on the stock of stores and the flow of new stores of MD and BK in each local market, as well as local market characteristics such as population, density, age distribution, average rent, income per capita, local retail taxes, and distance to the headquarters of the firm in UK.

We index firms by \( i \in \{BK, MD\} \), local markets by \( m \), and years by \( t \). The specification of the model is the one in the Example on Capacity Investment in an Oligopoly Industry in section 2.1 above. \( X_{imt} \) represents the number of installed stores of firm \( i \) in market \( m \) at the beginning of the year. The maximum value of \( X_{imt} \) in the sample is 13, and we consider that the set of possible values of \( X_{imt} \) is \( \{0, 1, ..., 15\} \). Therefore, the state space \( \mathcal{X} \) is \( \{0, 1, ..., 15\} \times \{0, 1, ..., 15\} \) that has 256 grid points. \( Y_{imt} \) is the binary indicator of the event "firm \( i \) opens a new store in market \( m \) at year \( t \)". For the code that we provide here, we consider that market characteristics are constant over time, and use market-specific mean values of these variables. However, it is straightforward to extend the code to accommodate exogenous state variables that evolve over time according to first order Markov processes. The measure of market size \( S_m \) is total population in the district. For some specifications, we allow the cost of investment to depend on market characteristics such as average rent, retail taxes, population density, or average income.

2.3 NPL Estimation

For an arbitrary vector of players' CCPs, \( P \equiv \{P_i(\mathbf{X}) : i = 1, 2; \mathbf{X} \in \mathcal{X}\} \), define the pseudo log-likelihood function:

\[
Q(\theta, \mathbf{P}) = \sum_{m=1}^{M} \sum_{i=1}^{2} \sum_{t=1}^{T} Y_{imt} \ln G_i(\left[\mathbf{z}_1^{P}(1) - \mathbf{z}_1^{P}(0)\right] \theta_i - \left[\mathbf{e}_1^{P}(1) - \mathbf{e}_1^{P}(0)\right]) \\
+ \ (1 - Y_{imt}) \ln \left(1 - G_i\left(\left[\mathbf{z}_1^{P}(1) - \mathbf{z}_1^{P}(0)\right] \theta_i - \left[\mathbf{e}_1^{P}(1) - \mathbf{e}_1^{P}(0)\right]\right)\right)
\]

(22)

4We would like to thank Otto Toivanen and Michael Waterson for generously sharing their data with us.
In this likelihood function, choice probabilities are best responses to an arbitrary $P$. The arbitrary probabilities in $P$ may be interpreted as players’ beliefs about other players’ expected behavior. These beliefs $P$ are parameters to estimate together with $\theta$. When $G_i$ is the logistic function (or the CDF of the standard normal), the function $Q(\theta, P)$ is the likelihood of a Logit (Probit) model where the parameter associated with the explanatory variable $e_{imt}(1) - e_{imt}(0)$ is restricted to be $-1$. For every possible value of $P$, the likelihood $Q(\theta, P)$ is globally concave in $\theta$. This property simplifies significantly the implementation of the NPL algorithm.

Let $P^0$ be the true vector of CCPs in the population under study, and let $\hat{P}^0$ be a nonparametric consistent estimator of $P^0$. For instance, a frequency estimator of $P^0$ is:

$$\hat{P}^0_i(X) = \frac{\sum_{m=1}^M \sum_{t=1}^T Y_{imt} 1\{X_{mt} = X\}}{\sum_{m=1}^M \sum_{t=1}^T 1\{X_{mt} = X\}}$$

The two-step estimator is defined as the value of $\theta$ that maximizes the pseudo likelihood $Q(\theta, \hat{P}^0)$. The estimator is consistent and asymptotically normal. Its main computational cost is in the calculation of the present values $\tilde{z}^P_{imt}$ and $\tilde{e}^P_{imt}$ following the procedure described in section 2.1. However, for the example we consider here the dimension of the state space $\mathcal{X}$ is small and the computation of these present values is quite simple. The main limitations of the two-step estimator are its asymptotic inefficiency, its large finite sample bias, and its problems to accommodate unobserved variables for the econometrician which are common knowledge to players, such as unobserved market characteristics.

If the equilibrium that generates the data is Lyapunov stable, then a recursive version of the two-step estimator, i.e., a K-step estimator, has better asymptotic and finite sample properties than the two-step estimator (see Aguirregabiria and Mira, 2007 and 2009, and Kasahara and Shimotsu, 2008). Given an initial nonparametric estimator $\hat{P}^0$, the sequence of K-step estimators $\{\hat{\theta}^K, \hat{P}^K : K \geq 1\}$ is defined as:

$$\hat{\theta}^K = \arg\max Q(\theta, \hat{P}^{K-1})$$

---

5 Let $P = \Psi(\theta, P)$ be the fixed point problem that defines an equilibrium for a given vector of structural parameters $\theta$. In our model, the equilibrium mapping $\Psi(\theta, P)$ is $G_i([z^0_{it}(1) - z^0_{it}(0)]\theta_i - [e^P_{it}(1) - e^P_{it}(0)])$ for every player $i$ and state $X$. An equilibrium $P^*$ is Lyapunov stable if the Jacobian matrix $\partial\Psi(\theta, P^*)/\partial P$ has a spectral radius smaller than 1. The spectral radius is the maximum absolute eigenvalue.
where the probabilities in $\hat{P}^K$ are updated using the recursive formula

$$
\hat{P}^K_i(X_t) = G_i \left( \left[ z_{it}^{\hat{P}^{K-1}}(1) - z_{it}^{\hat{P}^{K-1}}(0) \right] \hat{\theta}^K - \left[ c_{it}^{\hat{P}^{K-1}}(1) - c_{it}^{\hat{P}^{K-1}}(0) \right] \right)
$$

This recursive procedure is called the Nested Pseudo Likelihood (NPL) algorithm. The limit K-step estimator, as $K$ goes to infinity, is an NPL fixed point associated with the initial estimator $\hat{P}^0$. In general, an NPL fixed point $(\hat{\theta}_{NPL-FF}, \hat{P}_{NPL-FF})$ is defined by two conditions: (1) $\hat{\theta}_{NPL-FF} = \arg \max Q(\theta, \hat{P}_{NPL-FF})$; and (2) $\hat{P}_{NPL-FF}$ is an equilibrium of the model given $\hat{\theta}_{NPL-FF}$.

The model may have multiple NPL fixed points. If the equilibrium that generates the data is Lyapunov stable, then a NPL fixed point that is obtained by initializing the NPL algorithm with a consistent initial estimator $\hat{P}^0$ is consistent, asymptotically normal, and it has smaller asymptotic variance and finite sample bias than the two-step estimator. If a NPL fixed point is obtained by initializing the NPL algorithm with an inconsistent initial $\hat{P}^0$, then the NPL fixed point is not necessarily consistent. In that context, the NPL estimator is defined as the NPL fixed point with the largest value of the pseudo likelihood function. The NPL estimator is consistent, asymptotically normal, and it has smaller asymptotic variance and finite sample bias than the two-step estimator (Aguirregabiria and Mira, 2007).

The NPL method has been used in different applications of empirical games or single-agent dynamic decision models, such as models of entry in oligopoly markets (Aguirregabiria, Mira, and Roman, 2007, and Suzuki, 2008), plant turnover and productivity (Collard-Wexler, 2008, and Tomlin, 2009), supermarket pricing strategies (Ellickson and Misra, 2008, and Kano, 2006), dynamic games of competition between airline networks (Aguirregabiria and Ho, 2009), competition between real estate agents (Han and Hong, 2008), adoption of new technologies (Lenzo, 2007), land use and deforestation (De Pinto and Nelson, 2007 and 2009), quality competition between nursing homes (Lin, 2008), firm investment (Sanchez-Mangas, 2002), entry and competition in the religion industry (Walrath, 2008), demand of durable goods (Lorincz, 2005), or dynamic labor demand (Aguirregabiria and Alonso-Borrego, 2009), among others.
3 Main Program (npl_dyngame.prg)

The file npl_dyngame.prg contains the main program where all the primitives are specified and the different procedures are called. This program is divided into five parts.

PART 1: Specification of Some Constants.
The user should specify the values of the following constants and parameters.

<table>
<thead>
<tr>
<th>Program Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filedat</td>
<td>Name and address of the datafile.</td>
</tr>
<tr>
<td>nvar</td>
<td>Number of variables in the datafile.</td>
</tr>
<tr>
<td>nobs</td>
<td>Number of observations in the datafile.</td>
</tr>
<tr>
<td>mmarket</td>
<td>Number of markets in the dataset.</td>
</tr>
<tr>
<td>nyear</td>
<td>Number of years in the dataset.</td>
</tr>
<tr>
<td>maxstore</td>
<td>Maximum number of stores (i.e., max, value of $X_{it}$)</td>
</tr>
<tr>
<td>namesb1</td>
<td>Vector with names of parameters that vary across firms</td>
</tr>
<tr>
<td>namesb2</td>
<td>Vector with names of parameters that do not vary across firms</td>
</tr>
<tr>
<td>nplayer</td>
<td>Number of players</td>
</tr>
<tr>
<td>maxiter</td>
<td>Maximum number of iterations for the NPL algorithm</td>
</tr>
<tr>
<td>dfact</td>
<td>Discount factor parameter, $\beta$</td>
</tr>
</tbody>
</table>

Given this information, the program generates the matrix $vstate$ with all the possible values of the vector of state variables $X_t$. Each row of $vstate$ represents one value of $(X_{1mt}, X_{2mt})$.

PART 2: Reading Data and Construction of Vectors with observations of state and decision variables
The vectors $x_{bk}$ and $x_{md}$ contain the observations of the state variables $X_{1mt}$ and $X_{2mt}$. The vectors $a_{bk}$ and $a_{md}$ contain the observations of the decision variables $Y_{1mt}$ and $Y_{2mt}$.

PART 3: Procedures
This part of the program contains the different procedures called by the main program: freqprob, procedure for the initial estimates of CCPs; miprobit, procedure for the maximum likelihood estimation of a Probit model with constrains on parameters; and npl_bkmd, procedure for the NPL algorithm.

PART 4: Estimation of Initial Probabilities.
It calls the procedure \texttt{freqprob} for the frequency estimation of CCPs. In the version of the program that we provide here by default, initial CCPs are estimated separately market by market. Alternatively, the user could include time invariant market characteristics (i.e., population, average income, density, etc) as explanatory variables and call the procedure \texttt{freqprob} only once but including all the markets. The user could also prefer to use a Kernel estimator instead of the frequency estimator.

By running this part of the program, we get a vector of estimated initial CCPs called \texttt{prob_freq}. Alternatively, if we want to search for multiple NPL fixed points, we can replace this estimated vector by an arbitrary value of \texttt{prob_freq}. For instance, initialized the NPL algorithm with a vector of constant probabilities, e.g., \texttt{prob_freq = (1/2)*ones(nmarket*nstate,nplayer)}; or with random draws from a uniform distribution, e.g., \texttt{prob_freq = rndu(nmarket*nstate,nplayer)};

\textbf{PART 5: NPL Estimation.}

This part of the program calls the procedure for NPL estimation, \texttt{npl_bkmd}, that generates \texttt{maxiter} iterations of the NPL algorithm given an initial vector of CCPs.

\section{Procedures}

\subsection{freqprob}

This procedure obtains frequency or ’cell’ estimates of the probability distribution of a vector of discrete random variables $Y$ conditional on a vector of discrete random variables $X$. 
**Procedure freqprob**

Format: \{ prob \} = freqprob(yobs, xobs, xval)

<table>
<thead>
<tr>
<th>Name</th>
<th>INPUT VARIABLES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yobs</td>
<td>((nobs \times q)) matrix with sample observations of (Y = Y_1 \sim Y_q)</td>
<td></td>
</tr>
<tr>
<td>xobs</td>
<td>((nobs \times k)) matrix with sample observations of (X = X_1 \sim X_q)</td>
<td></td>
</tr>
<tr>
<td>xval</td>
<td>((numx \times k)) matrix with values of (X) at which we want to estimate the conditional probability function (Pr(Y</td>
<td>X)).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>OUTPUT VARIABLES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>((numx \times q)) matrix with estimates of (Pr(Y</td>
<td>X)) for every value of (X) in (xval) (Pr(Y_1 = 1</td>
</tr>
</tbody>
</table>

Frequency estimators for empty cells (i.e., values in \(xval\) for which there are zero observations in \(xobs\)) are defined to be zero. This frequency or 'cell' estimator of the conditional probability of discrete random variables is consistent under very weak conditions.

### 4.2 miprobit

This procedure obtains the Maximum Likelihood estimates (MLE) of a binary Probit model. The parameters of some explanatory variables can be restricted to take specific values. The algorithm to obtain the MLE is Newton’s method with analytical expressions for gradient and Hessian. The log-likelihood function of this Probit model is globally concave in the parameters. Therefore, in the absence of multicollinearity problems or other numerical issues (e.g., choice probabilities too close to zero or one), Newton’s algorithm always converges to the MLE regardless the value of the parameters that we use to initialize the algorithm.
Procedure miprobit
Format: \{best, varest, llike\} = miprobit(ydum, x, rest, b0, nombres, out)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ydum</td>
<td>((nobs \times 1)) vector with sample observations of the dependent variable</td>
</tr>
<tr>
<td>x</td>
<td>((nobs \times K)) matrix with sample observations of explanatory variables associated with the unrestricted parameters</td>
</tr>
<tr>
<td>rest</td>
<td>((nobs \times 1)) vector with observations of the sum of the explanatory variables whose parameters are restricted to be 1</td>
</tr>
<tr>
<td>b0</td>
<td>((K \times 1)) vector with values of parameters to initialized Newton’s method</td>
</tr>
<tr>
<td>nombres</td>
<td>((K \times 1)) vector with names of parameters to estimate</td>
</tr>
<tr>
<td>out</td>
<td>Binary scalar that specifies screen output: 0 = no table of results; 1 = table with estimation results</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>((K \times 1)) vector with maximum likelihood estimates</td>
</tr>
<tr>
<td>varest</td>
<td>((K \times K)) matrix with estimated variances-covariances of estimates</td>
</tr>
<tr>
<td>llike</td>
<td>Scalar with value of log-likelihood function at the MLE</td>
</tr>
</tbody>
</table>

4.3 npl_bkmd

This procedure iterates in the Nested Pseudo Likelihood algorithm given an initial vector of CCPs. At each NPL iteration, the algorithm performs three main tasks.

Task 1: Computing the matrices \(ztilda_{bk}\) and \(ztilda_{md}\) and the vectors \(etilda_{bk}\) and \(etilda_{md}\) for every market and every sample observation. From a computational point of view, this is the most demanding part of an NPL iteration. The update of these matrices and vectors is done market by market because each market has its own CCPs. This task is divided into three sub-tasks: (a) construction of one-period expected profits; (b) construction of transition probabilities; and (c) computation of \(ztilda_{bk}\), \(ztilda_{md}\), \(etilda_{bk}\), and \(etilda_{md}\).
**Task 2:** Call to the procedure `miprobit` for the pseudo maximum likelihood estimation of structural parameters.

**Task 3:** Update of the CCPs.

---

**Procedure npl_bkmd**

Format: `{thetaest,varest,pchoice,pest_obs} = npl_bkmd(yobs,xobs,msize,zmarket,pchoice,mstate,beta,kiter,namesb)`

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT VARIABLES</strong></td>
<td></td>
</tr>
<tr>
<td>yobs</td>
<td>((nobs \times 2)) matrix with sample observations of players’ choice variables</td>
</tr>
<tr>
<td>xobs</td>
<td>((nobs \times 2)) matrix with sample observations of players’ endogenous state variables</td>
</tr>
<tr>
<td>msize</td>
<td>((nmarket \times 1)) vector with sample observations of market size (population)</td>
</tr>
<tr>
<td>zmarket</td>
<td>((nmarket \times Kz)) matrix with sample observations of time-invariant market characteristics</td>
</tr>
<tr>
<td>pchoice</td>
<td>((nstate \times nmarket \times 2)) matrix with initial vector of CCPs for each market, each state, and each player</td>
</tr>
<tr>
<td>mstate</td>
<td>((nstate \times 2)) matrix with all the possible values of the endogenous state variables</td>
</tr>
<tr>
<td>beta</td>
<td>Scalar with value of the discount factor</td>
</tr>
<tr>
<td>kiter</td>
<td>Scalar natural number with number of NPL iterations</td>
</tr>
<tr>
<td>namesb</td>
<td>((K \times 1)) vector with names of the structural parameters to estimate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>OUTPUT VARIABLES</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thetaest</td>
<td>((K \times 1)) vector with parameter estimates at the last NPL iteration</td>
</tr>
<tr>
<td>varest</td>
<td>((K \times K)) matrix of variances and covariances</td>
</tr>
<tr>
<td>pest</td>
<td>((nstate \times nmarket \times 2)) matrix with estimates of CCPs for every market, state and player</td>
</tr>
<tr>
<td>pest_obs</td>
<td>((nobs \times 2)) matrix with estimates of CCPs for every observations and player</td>
</tr>
</tbody>
</table>
5 Output and Empirical Results

For each NPL iteration, the code provides the following screen output:

- At every Pseudo ML (PML) iteration (within an NPL iteration). Index of the PML iteration; value of pseudo log-likelihood; value of the criterion for convergence, \( \max_q |\hat{\theta}_q^j - \hat{\theta}_q^{j-1}| \), where \( q \) represents the index for the PML iteration. For instance:

  Pseudo MLE Iteration = 2.000  
  Log-Likelihood function = -2883.  
  Criterion = 1.595

- At the end of every NPL iteration.

  Total number of PML iterations; Final value of the pseudo log-likelihood; Likelihood Ratio Index (measure of goodness-of-fit); Pseudo R-square; Parameter estimates and asymptotic standard errors; Number of NPL iterations and value of the NPL convergence criterion, i.e., \( \max_j |\hat{\theta}_j^K - \hat{\theta}_j^{K-1}| \) where \( K \) represents the index for the NPL iteration. For instance:

  Number of Iterations = 10.00  
  Log-Likelihood function = -655.7  
  Likelihood Ratio Index = 0.3699  
  Pseudo-R2 = 0.3228

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t-ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP0_BK</td>
<td>0.5849</td>
<td>0.1077</td>
<td>5.430</td>
</tr>
<tr>
<td>VP1_BK</td>
<td>-0.2096</td>
<td>0.0552</td>
<td>-3.792</td>
</tr>
<tr>
<td>VP2_BK</td>
<td>-0.0110</td>
<td>0.0029</td>
<td>-3.761</td>
</tr>
<tr>
<td>FC0_BK</td>
<td>0.0784</td>
<td>0.0213</td>
<td>3.674</td>
</tr>
<tr>
<td>FC1_BK</td>
<td>0.0790</td>
<td>0.0445</td>
<td>1.775</td>
</tr>
<tr>
<td>FC2_BK</td>
<td>-0.0078</td>
<td>0.0059</td>
<td>-1.333</td>
</tr>
<tr>
<td>VP0_MD</td>
<td>0.8303</td>
<td>0.2968</td>
<td>2.798</td>
</tr>
<tr>
<td>VP1_MD</td>
<td>-0.0024</td>
<td>0.0392</td>
<td>-0.0615</td>
</tr>
<tr>
<td>VP2_MD</td>
<td>0.0008</td>
<td>0.0027</td>
<td>0.3184</td>
</tr>
<tr>
<td>FC0_MD</td>
<td>0.0822</td>
<td>0.0332</td>
<td>2.473</td>
</tr>
<tr>
<td>FC1_MD</td>
<td>0.1076</td>
<td>0.0400</td>
<td>2.689</td>
</tr>
<tr>
<td>FC2_MD</td>
<td>-0.0034</td>
<td>0.0023</td>
<td>-1.435</td>
</tr>
<tr>
<td>DENSITY</td>
<td>10.74</td>
<td>2.817</td>
<td>3.811</td>
</tr>
<tr>
<td>GDP</td>
<td>0.0003</td>
<td>0.0002</td>
<td>1.374</td>
</tr>
<tr>
<td>RENT</td>
<td>-0.0016</td>
<td>0.0006</td>
<td>-2.606</td>
</tr>
<tr>
<td>TAX</td>
<td>-1.746e-005</td>
<td>5.489e-005</td>
<td>-0.3181</td>
</tr>
</tbody>
</table>

NPL ITERATION = 2.000 Criterion = 10.74
The following table contains estimates using the McDonalds-Burger King dataset and the model described in section 2.

<table>
<thead>
<tr>
<th></th>
<th>Two Step Estimates</th>
<th>NPL Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Burger King</td>
<td>McDonalds</td>
</tr>
<tr>
<td></td>
<td>Burger King</td>
<td>McDonalds</td>
</tr>
<tr>
<td>Variable Profits:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\theta_0^{VP}$</td>
<td>0.5849 (0.1077)*</td>
<td>0.8303 (0.2968)*</td>
</tr>
<tr>
<td>$\theta_1^{VP}$</td>
<td>-0.2096 (0.0552)*</td>
<td>-0.0024 (0.0392)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Costs:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\theta_0^{FC}$</td>
<td>0.0784 (0.0213)*</td>
<td>0.0822 (0.0332)*</td>
</tr>
<tr>
<td>$\theta_1^{FC}$</td>
<td>0.0790 (0.0420)*</td>
<td>0.1076 (0.0400)*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-Likelihood</td>
<td>-655.7</td>
<td>-893.4</td>
</tr>
<tr>
<td>Distance $</td>
<td></td>
<td>P^K - P^{K-1}</td>
</tr>
<tr>
<td># NPL iterations</td>
<td>1</td>
<td>31</td>
</tr>
</tbody>
</table>
(a) Convergence: The NPL fixed point reported in this table is the one that we converge to when the NPL algorithm is initialized with the nonparametric frequency estimator. It takes 31 NPL iterations to converge to this fixed point. Figure 1 presents the NPL convergence criterion, \(\max_j |\hat{\theta}_j^K - \hat{\theta}_j^{K-1}|\), at every NPL iteration. We present the convergence criterion both in levels and in logarithms because the representation in logarithms provides a better picture of the convergence rate. Notice that convergence is not monotonic. This is because the consistent NPL fixed point is just a local contraction, and not a global contraction. In fact, as mentioned above, convergence of the NPL algorithm is not guaranteed. In particular, it is possible that the algorithm converges to a cycle of two or more than two values of \(\hat{\theta}\). In that case, NPL iterations can be combined with techniques to deal with cycles in fixed point iteration algorithms. For instance, if we find a cycle, a possible solution is to re-start the NPL algorithm using as initial probabilities the mean values of the probabilities in the different point of the cycle. This is a very simple but it tends to be very effective.

Figure 1

(b) NPL Estimator (Global search): The NPL fixed point associated with the initial frequency estimator is consistent. However, it is not necessarily the NPL estimator because there may be other NPL fixed points with higher value of the pseudo likelihood function. We have implemented the NPL algorithm for different values of the initial \(P_0\): e.g., \(P_0 = 0.5\) for every market, player and state; \(P_0 = 0.0000001\) for every market, player and state; \(P_0 = 0.999999\) for every market, player and state; and \(P_0 = \) vector of random draws from \(Uniform(0, 1)\). In this application, and for all the initial values we have tried, we have converge always to the same NPL fixed point. The
convergence path is quite similar in all the cases we have tried. Figure 2 presents the convergence paths for three cases: $P_0 = 0.5$, $P_0 = 0.000001$, and $P_0 = 0.999999$.

![Figure 2](image)

Note that convergence to the NPL fixed point is quite slow. In fact, as shown by the figures in logarithms, the rate of convergence declines when we approach to the NPL fixed point. Regardless the vector of CCPs that we use to initialize the NPL algorithm, 15 NPL iterations or less take us very close to an NPL fixed point. However, it takes other 15 iterations to really converge to that fixed point. The convergence criterion that we use is $\max_j |\hat{\theta}_j^K - \hat{\theta}_j^{K-1}| < 10^{-6}$. We might want to
relax that convergence criterion. Alternatively, it is possible to use "accelerate NPL iterations" as proposed by Kasahara and Shimotsu (2008). For instance, we can apply more than one iteration in the best response mapping at each NPL iteration. Iterations in the best response mapping are computationally costly, but this additional cost might be compensated by a smaller number of NPL iterations.

(c) Comparing Two-Step and NPL estimators. In this application, we find important differences between the parameter estimates using two-step and NPL methods. Figures 3 presents the estimated variables profit functions and fixed cost functions for BK and MD under the two estimation methods.
Appendix: Gauss code

// *********************************************************************
// NPL_DYNAGAME_150909.prg
//
// THIS PROGRAM ESTIMATES A DYNAMIC GAME OF ENTRY-EXIT USING
// THE NESTED PSEUDO LIKELIHOOD (NPL) METHOD, AND ACTUAL DATA
// ON MCDONALDS AND BURGER KING LOCATION OF OUTLETS IN UK
//
// by VICTOR AGUIRREGABIRIA
//
// SEPTEMBER 2009
//
// *********************************************************************

// SPECIFICATION OF ONE-PERIOD PROFIT FUNCTION
// The profit function for firm i is:
//
// Ui = zi(ai,aj) * thetai - ai * epsi
//
// where ai is the new entry decision of firm i, aj is the
// new entry decision of firm j, zi(ai,aj) are vectors
// of variables, and thetai is a vector of parameters. More specifically,
//
// thetai = (VP0i, VP1i, VP2i, FC1i, FC2i)
//
// where VPOi, VP1i, and VP2i are parameters in the variable profit function,
// FC1i and FC2i are parameters in fixed costs. And
//
// zi(ai,aj) = { S * 1(xi + ai > 0) }
// ~{ S * (xi + ai - xj - aj) }
// ~{ S * (xi + ai - xj - aj)^2 }
// ~{ -1(xi + ai > 0) }
// ~{ -(xi + ai) }
// ~{ -(xi + ai)^2 }
//
// new;
closeall;
library pgraph gauss;
format /mb1 /ros 16,4;
// *********************************************************************
// PART 1: SPECIFICATION OF SOME CONSTANTS
//*********************************************************************
// Constants of the datafile
// Name and address of data file
filedat = "c:\mypapers\arvind_rationalizability\data\toivanen_waterson_nolondon_120809.dat";
nobs = 2110; // Number of observations in data file
nmarket = 422; // Number of local markets
nyear = 5; // Number of years
nvar = 27; // Number of variables in dataset
// Constants of the model
maxstore = 15; // maximum number of stores
nplayer = 2;
dfact = 0.95 ; // Discount factor
maxiter = 50 ;
namesb1 = "VP0_BK" | "VP1_BK" | "VP2_BK" | "FC0_BK" | "FC1_BK" | "FC2_BK"
 | "VP0_MD" | "VP1_MD" | "VP2_MD" | "FC0_MD" | "FC1_MD" | "FC2_MD" ;
namesb2 = "DENSITY" | "GDP" | "RENT" | "TAX" ;
namesb = namesb1 | namesb2 ;
// Calculating some constants
vstate = seqa(0,1,maxstore) ;
vstate = (vstate.*.ones(maxstore,1)) ~(ones(maxstore,1).*vstate) ;
// Matrix with all possible values of the state variables
nstate = rows(vstate) ;
kp1 = rows(namesb1)/2 ;
kp2 = rows(namesb2) ;
kparam = rows(namesb) ;
// ***************************************************
// PART 2. READING DATA AND CONSTRUCTION OF VARIABLES
// ***************************************************
open dtin = ^filedat for read varindxi ;
data = readr(dtin,nobs);
dtin = close(dtin) ;
county_name = data[.,1] ;
district_name = data[.,2] ;
county_code = data[.,3] ;
district_code = data[.,4] ;
year = data[.,5] ;
mcd_stock = data[.,6] ;
mcd_entry = data[.,7] ;
mcd_entdum = data[.,8] ;
bk_stock = data[.,9] ;
bk_entry = data[.,10] ;
bk_entdum = data[.,11] ;
district_area = data[.,12] ;
population = data[.,13] ;
pop_0514 = data[.,14] ;
pop_1529 = data[.,15] ;
pop_4559 = data[.,16] ;
pop_6064 = data[.,17] ;
pop_6574 = data[.,18] ;
avg_rent = data[.,19] ;
ctax = data[.,20] ;
ecac = data[.,21] ;
ue = data[.,22] ;
gdp_pc = data[.,23] ;
dist_bkhq_miles = data[.,24] ;
dist_bkhq_minu = data[.,25] ;
dist_mdhq_miles = data[.,26] ;
dist_mdhq_minu = data[.,27] ;
// Construction of variables
x_bk = bk_stock ; // Stock of stores for BK
x_md = mcd_stock ; // Stock of stores for MD
a_bk = (bk_entry.>0) ; // Dummy of new entry for BK
a_md = (mcd_entry.>0) ; // Dummy of new entry for MD
population = population/1000 ; // Population in millions
density = population./district_area ;

22
// Market specific mean values of some exogenous explanatory variables
marketsize = meanc(reshape(population,nmarket,nyear)')
zmarket = meanc(reshape(density,nmarket,nyear)')
~meanc(reshape(gdp_pc,nmarket,nyear)')
~meanc(reshape(avg_rent,nmarket,nyear)')
~meanc(reshape(ctax,nmarket,nyear)')

// ************************************
// PART 3. PROCEDURES
// ************************************
// --------------------------------------
// A. PROCEDURE for FREQUENCY ESTIMATOR
// --------------------------------------
proc (1) = freqprob(yobs,xobs,xval) ;
// FREQPORB.SRC Procedure that obtains a frequency estimation
// of Prob(Y|X) where Y is a vector of binary
// variables and X is a vector of discrete variables
// FORMAT:
// freqp = freqprob(yobs,xobs,xval)
// INPUTS:
// yobs - (nobs x q) vector with sample observations
// of Y = Y1 ~Y2 ~... ~Yq
//
// xobs - (nobs x k) matrix with sample observations of X
//
// xval - (numx x k) matrix with the values of X for which
// we want to estimate Prob(Y|X).
// OUTPUTS:
// freqp - (numx x q) vector with frequency estimates of
// Pr(Y|X) for each value in xval.
// Pr(Y1=1|X) ~Pr(Y2=1|X) ~... ~Pr(Yq=1|X)
// ----------------------------------------------
// B. PRECEDURE for CONSTRAINED PROBIT ESTIMATOR
// ----------------------------------------------
proc (3) = miprobit(ydum,x,rest,b0,nombres,out) ;
// MIPROBIT - Estimation of a Probit Model by Maximum Likelihood
// The optimization algorithm is a Newton's method
// with analytical gradient and hessian

// FORMAT {best,varest,llike} = miprobit(ydum,x,rest,b0,nombres,out)

// INPUTS
// ydum - (nobs x 1) vector with observations of the dependent variable
// x - (nobs x k) matrix with observations of explanatory variables
// associated with the unrestricted parameters
// rest - vector with observations of the sum of the explanatory
// variables whose parameters are restricted to be 1
// (Note that the value 1 is without loss of generality
// if the variable rest is constructed appropriately)
// b0 - (k x 1) vector with values of parameters to initialized
// Newton's methods
// nombres - (k x 1) vector with names of parameters to estimate
// out - 0=no table of results; 1=table with estimation results

// OUTPUTS
// best - ML estimates
// varest - estimate of the covariance matrix
// llike - value of log-likelihood function at the MLE

local myzero, nobs, nparam, eps, iter, llike, criter, Fxb0, phixb0, lamdab0, dlogLb0, d2logLb0, b1, lamda0, lamda1, Avarb, sdb, tstat, numy1, numy0, logL0, LRI, pseudoR2, k ;
myzero = 1e-36 ;
nobs = rows(ydum) ;
nparam = cols(x) ;
eps = 1E-6 ;
iter=1 ;
llike = 1000 ;
criter = 1000 ;
do while (criter>eps) ;
if (out==1) ;
" Pseudo MLE Iteration = " iter ;
"Log-Likelihood function = " llike ;
"Criterion = " criter ;
"" ;
endif ;
Fxb0 = cdfn(x*b0+rest) ;
Fxb0 = Fxb0 + (myzero - Fxb0).*(Fxb0.<myzero) + (1-myzero - Fxb0).*(Fxb0.(1-myzero)) ;
llike = ydum'*ln(Fxb0) + (1-ydum)'*ln(1-Fxb0) ;
phixb0 = pdfn(x*b0+rest) ;
lamdab0 = ydum.*phixb0./Fxb0 + (1-ydum).*(-phixb0./((1-Fxb0)) ;
dlogLb0 = x’*lamdab0 ;
d2logLb0 = -(lamdab0.*(lamdab0 + x*b0 + rest)).*x’ ;
b1 = b0 - inv(d2logLb0)*dlogLb0 ;
criter = maxc(abs(b1-b0)) ;

b0 = b1;
iter = iter + 1;
endo;
Fxb0 = cdfn(x*b0 + rest);
Fxb0 = Fxb0 + (myzero - Fxb0).*\(1-Fxb0\) + (1-myzero - Fxb0).*\(1-Fxb0\);
llike = ydum'ln(Fxb0) + (1-ydum)'ln(1-Fxb0);
phixb0 = pdfn(x*b0 + rest);
lamda0 = -phixb0/(1-Fxb0);
lamda1 = phixb0/Fxb0;
Avarb = ((lamda0.*lamda1).*x)'*x;
Avarb = inv(-Avarb);
sdb = sqrt(diag(Avarb));
tstat = b0./sdb;
numy1 = sumc(ydum);
numy0 = nobs - numy1;
logL0 = numy1*ln(numy1) + numy0*ln(numy0) - nobs*ln(nobs);
LRI = 1 - llike/logL0;
pseudoR2 = 1 - \((ydum - Fxb0)'*(ydum - Fxb0)\)/numy1;
if (out==1);
"Number of Iterations = " iter;
"Log-Likelihood function = " llike;
"Likelihood Ratio Index = " LRI;
"Pseudo-R2 = " pseudoR2;
"
"------------------------------------------------------------------"
" Parameter Estimate Standard t-ratios"
" Errors"
"------------------------------------------------------------------"
k=1;
do while k<=nparam;
print $nombres[k];;b0[k];;sdb[k];;tstat[k];
k=k+1;
endo;
endif;
retp(b0,Avarb,llike);
endp;

// C. PROCEDURE for NPL ESTIMATOR
// ---------------------------------
proc (4) = npl_bkmd(yobs, xobs, msize, zmarket, pchoice, mstate, beta, kiter, namesb);
// ---------------------------------
// NPL_BKMD
// This procedure iterates in the NPL algorithm given an initial vector
// of CCPs. The model is the dynamic game of local market entry for
// McDonalds and Burger King. The procedure returns the vector of parameter
// estimates, the variance matrix, and the matrix with players choice
// probabilities at every state, and at every sample point.
// FORMAT (thetaest,varest,pest,pest_obs) =
// npl_bkmd(yobs, xobs, msize, zmarket, pchoice, mstate, beta, kiter, namesb)
// INPUTS
// yobs - (nobs x 2) matrix with observations of players’ choices
// xobs - (nobs x 2) matrix with observations of players’ endogenous
// state variables
// msize - (nmarket x 1) vector with observations of market size (population)
// zmarket - (nmarket x kz) matrix with observations of time-invariant
// market characteristics
// pchoice - (nstate*nmarket x 2) matrix with initial vector of CCPs for
// every market, state and player
// mstate - (nstate x 2) matrix with all the possible values of the
// endogenous state variables
// beta - Scalar with value of the discount factor
// kiter - Scalar natural number with number of NPL iterations
// namesb - (K x 1) vector with names of the structural parameters

// OUTPUTS
// thetaest - (K x1) vector with parameter estimates at the last NPL iteration
// varest - (K xK) matrix of variances and covariances
// pest - (nstate*nmarket x 2) matrix with estimates of CCPs for
// every market, state and player
// pest_obs - (nobs x 2) matrix with estimates of CCPs for every
// observation and state
// obervation and state

local myzero, nobs, nmarket, nyear, nplayer, ns, numx, ktot, kvpfc, kz, indxobs, j,
xbk, xmd, npliter, criterion, conv_const, theta0,
p_bk, p_md, ztilda_bk, ztilda_md, etilda_bk, etilda_md,
ztilda_obs_bk, ztilda_obs_md, etilda_obs_bk, etilda_obs_md,
m, valmsize, valzmarket,
zbk_00, zbk_01, zbk_10, zbk_11, zmd_00, zmd_01, zmd_10, zmd_11,
eprofbk_0, eprofbk_1, eprofmd_0, eprofmd_1,
tranxbk_bk0, tranxbk_bk1, tranxmd_md0, tranxmd_md1, tranxmd_bk, tranxbk_md,
tottran_bk0, tottran_bk1, tottran_md0, tottran_md1, uncontran,
value_z_bk, value_z_md, value_e_bk, value_e_md,
zt_bk, zt_md, et_bk, et_md, count1, count2,
zobs, eobs, thetaest, varest, likelihood, theta_bk, theta_md, pest_obs ;

// ---------------
// Some constants
// ---------------
myzero = 1e-12 ; // Constant for truncation of CCPs to avoid numerical errors
nobs = rows(yobs) ; // Total number of market*year observations
nmarket = rows(msize) ; // Total number of markets in the sample
nyear = nobs/nmarket ; // Number of years in the sample (balanced panel)
if nyear/=int(nyear) ; "ERROR: Number of years is not an integer"; end; endif;
nplayer = cols(yobs) ;
ns = rows(pchoice)/nmarket ; // number of states in a single market
if ns/=int(ns) ; "ERROR: Number of states in a single market is not an integer";
end; endif;
umx = sqrt(ns) ; // number of values of xbk or xmd
if numx/=int(numx) ; "ERROR: Number of values of xbk or xmd is not an integer";
end; endif;
ktot = rows(namesb) ; // Total number of parameters
kz = cols(zmarket);  // Number of parameters associated with the control variables in zmarket
kvafc = (ktot-kz)/2;  // Number of parameters in var profits and fixed costs for a single firm
if kvafc/=int(kvafc); "ERROR: Number of parameters in var profits and fixed costs is not an integer"; end; endif;
xbk = mstate[.,1];  // vector stock of stores for BK
xmd = mstate[.,2];  // vector stock of stores for MD
indxobs = zeros(nobs,1);
j=1;
do while j<=ns;
    indxobs = indxobs + j.*prodc((xobs==mstate[j,.])');
j=j+1;
endo;
// --------------
// NPL algorithm
// --------------
criterion = 1000;
conv_const = 1e-6;
theta0 = zeros(ktot,1);
npliter=1;
do while (npliter<=kiter).and(criterion>conv_const);
    "NPL ITERATION ="; npliter ; 
    "Criterion ="; criterion ;
    ";
endo;
// TASK 1: Computing the matrices ztilda_bk and ztilda_md and the vectors etilda_bk and etilda_md
// for every market and every sample observation
ztilda_bk = zeros(nmarket*ns,kvafc+kz);
ztilda_md = zeros(nmarket*ns,kvafc+kz);
etilda_bk = zeros(nmarket*ns,1);
etilda_md = zeros(nmarket*ns,1);
ztilda_obs_bk = zeros(nobs,kvafc+kz);
ztilda_obs_md = zeros(nobs,kvafc+kz);
etilda_obs_bk = zeros(nobs,1);
etilda_obs_md = zeros(nobs,1);
m=1;
do while m<=nmarket;
    valmsize = msize[m];
    valzmarket = zmarket[m,] ;
    p_bk = pchoice((m-1)*ns+1:m*ns,1);
p_md = pchoice((m-1)*ns+1:m*ns,2);
p_bk = (p_bk.<myzero).*myzero
    + (p_bk.>=1-myzero).*((1-myzero)) ;
p_md = (p_md.<myzero).*myzero
27
\[ (p_{md} \geq (1-myzero)).(1-myzero) + (p_{md} = myzero).*(p_{md} < (1-myzero)).*p_{md}; \]

// Vectors of expected profits

zbk_00 = (valmsize.*(xbk.>0)) ~(valmsize.*(xbk-xmd))
~(valmsize.*(xbk-xmd)).*(xbk-xmd))
~(xbk.>0) ~(-xbk) ~(-xbk.*xbk)
~(valzmarket.*xbk);

zbk_01 = (valmsize.*(xbk.>0)) ~(valmsize.*(xbk-xmd-1))
~(valmsize.*(xbk-xmd-1)).*(xbk-xmd-1))
~(-((xbk+1).>0)) ~(xbk+1) ~(-(xbk+1).*(xbk+1))
~(valzmarket.*(xbk+1));

zmd_00 = (valmsize.*(xmd.>0)) ~(valmsize.*(xmd-xbk))
~(valmsize.*(xmd-xbk)).*(xmd-xbk))
~(xmd.>0) ~(xmd) ~(xmd.*xmd)
~(valzmarket.*xmd);

zmd_01 = (valmsize.*(xmd.>0)) ~(valmsize.*(xmd-xbk-1))
~(valmsize.*(xmd-xbk-1)).*(xmd-xbk-1))
~(-((xmd+1).>0)) ~(xmd+1) ~(-(xmd+1).*(xmd+1))
~(valzmarket.*(xmd+1));

eprofbk_0 = (1-p_md).*zbk_00 + p_md.*zbk_01; // Expected Profit BK if a=0
eprofbk_1 = (1-p_md).*zbk_10 + p_md.*zbk_11; // Expected Profit BK if a=1
eprofmd_0 = (1-p_bk).*zmd_00 + p_bk.*zmd_01; // Expected Profit MD if a=0
eprofmd_1 = (1-p_bk).*zmd_10 + p_bk.*zmd_11; // Expected Profit MD if a=1

// Transition probabilities

// Remember: vstate = xbk ~xmd
// where: xbk = (1|2| ... |14).*.(1|1|...|1)
// xmd = (1|1| ... |1) .*.(1|2|...|14)

// Transition xbk for BK: abk = 0
// Pr( xbk' | xbk, xmd, abk =0) = 1{xbk' = xbk)
tranxbk_bk0 = eye(numx); // Transition of xbk in the space of xbk
tranxbk_bk0 = tranxbk_bk0.*.ones(numx,numx); // Transition of xbk in the space of xbk, xmd
// Transition xbk for BK: abk = 1
// Pr( xbk' | xbk, xmd, abk =0) = 1{xbk' = xbk+1)
tranxbk_bk1 = (zeros(numx-1,1) ~eye(numx-1))
| (zeros(1,numx-1) ~1); // Transition of xbk in the space of xbk
tranxbk_bk1 = tranxbk_bk1.*.ones(numx,numx); // Transition of xbk in the space
of xbk, xmd
// Transition xmd for MD: amd = 0
// Pr( xmd' | xbk, xmd, amd =0) = 1{xmd' = xmd)
tranxmd_md0 = eye(numx); // Transition of xmd in the space of xmd
tranxmd_md0 = ones(numx,numx).*tranxmd_md0; // Transition of xmd in the space
of xbk, xmd
// Transition xmd for MD: amd = 1
// Pr( xmd' | xbk, xmd, amd =1) = 1{xmd' = xmd+1)
tranxmd_md1 = (zeros(numx-1,1) ~eye(numx-1))
| (zeros(1,numx-1) ~1); // Transition of xmd in the space of xmd
tranxmd_md1 = ones(numx,numx).*tranxmd_md1; // Transition of xmd in the space
of xbk, xmd
// Transition xmd from the point of view of BK who doesn’t know amd
// Pr( xmd' | xbk, xmd) = (1-p_md) * 1{xmd' = xmd) + p_md * 1{xmd' = xmd+1)
tranxmd_bk = (1-p_md).* tranxmd_md0 + p_md.* tranxmd_md1 ; // Transition of xmd in the point of view of BK
// Transition xmd from the point of view of MD who doesn’t know abk
// Pr( xbk' | xbk, xmd) = (1-p_bk) * 1{xbk' = xbk) + p_bk * 1{xbk' = xbk+1)
tranxbk_md = (1-p_bk).* tranxbk_bk0 + p_bk.* tranxbk_bk1 ; // Transition of xmd in the point of view of MD
// Total transition matrix of (xbk,xmd) for BK if abk = 0
// Pr( xbk',xmd' | xbk, xmd, abk=0) = Pr( xbk' | xbk, abk =0) * Pr( xmd' | xbk, xmd)
tottran_bk0 = tranxbk_bk0 .* tranxmd_md0 ; // Transition of (xbk,xmd) in the space of BK
if sumc(sumc(tottran_bk0').>
(1.00001)) or sumc(sumc(tottran_bk0').<
(0.99999))
"ERROR: Transition matrix does not sum 1" ; end ;
endif ;
// Total transition matrix of (xbk,xmd) for BK if abk = 1
// Pr( xbk',xmd' | xbk, xmd, abk=1) = Pr( xbk' | xbk, abk =1) * Pr( xmd' | xbk, xmd)
tottran_bk1 = tranxbk_bk1 .* tranxmd_md1 ; // Transition of (xbk,xmd) in the space of BK
if sumc(sumc(tottran_bk1').>
(1.00001)) or sumc(sumc(tottran_bk1').<
(0.99999))
"ERROR: Transition matrix does not sum 1" ; end ;
endif ;
// Total transition matrix of (xbk,xmd) for MD if amd = 0
// Pr( xbk',xmd' | xbk, xmd, amd=0) = Pr( xmd' | xmd, amd =0) * Pr( xbk' | xbk, xmd)
tottran_md0 = tranxmd_md0 .* tranxbk_md ; // Transition of (xbk,xmd) in the space of MD
if sumc(sumc(tottran_md0').>
(1.00001)) or sumc(sumc(tottran_md0').<
(0.99999))
"ERROR: Transition matrix does not sum 1" ; end ;
endif ;
// Total transition matrix of (xbk,xmd) for MD if amd = 1
// Pr( xbk',xmd' | xbk, xmd, amd=1) = Pr( xmd' | xmd, amd =1) * Pr( xbk' | xbk, xmd)
tottran_md1 = tranxmd_md1 .* tranxbk_md ; // Transition of (xbk,xmd) in the space of MD
if sumc(sumc(tottran_md1').>
(1.00001)) or sumc(sumc(tottran_md1').<
(0.99999))
"ERROR: Transition matrix does not sum 1" ; end ;
endif ;
// Unconditional transition matrix
uncontran = (1-p_bk) .* tottran_bk0 + p_bk .* tottran_bk1 ;
if sumc(sumc(uncontran')).>(1.00001) or sumc(sumc(uncontran')).<(0.99999)) ;
"ERROR: Transition matrix does not sum 1" ; end ;
endif ;
// -----------------------------------------------------------------------
// ztilda_bk, ztilda_md, etilda_bk, etilda_md for every possible state
// -----------------------------------------------------------------------
uncontran = inv(eye(ns) - beta*uncontran) ; // Matrix (I - beta*F)^-1
value_z_bk = (1-p_bk).*eprofbk_0 + p_bk.*eprofbk_1 ;
value_z_bk = uncontran * value_z_bk ; // Value Z function BK
value_e_bk = uncontran * pdfn(cdfni(p_bk)) ; // Value e function BK
value_z_md = (1-p_md).*eprofmd_0 + p_md.*eprofmd_1 ;
value_z_md = uncontran * value_z_md ; // Value Z function MD
value_e_md = uncontran * pdfn(cdfni(p_md)) ; // Value e function MD
zt_bk = (eprofbk_1 - eprofbk_0) + beta*(tottran_bk1-tottran_bk0)*value_z_bk ;
zt_md = (eprofmd_1 - eprofmd_0) + beta*(tottran_md1-tottran_md0)*value_z_md ;
et_bk = beta*(tottran_bk1-tottran_bk0)*value_e_bk ;
et_md = beta*(tottran_md1-tottran_md0)*value_e_md ;
// ------------
// Filling
// ------------
count1 = (m-1)*ns + 1 ;
count2 = m*ns ;
ztilda_bk[count1:count2,..] = zt_bk ;
ztilda_md[count1:count2,..] = zt_md ;
etilda_bk[count1:count2,..] = et_bk ;
etilda_md[count1:count2,..] = et_md ;
count1 = (m-1)*nyear + 1 ;
count2 = m*nyear ;
ztilda_obs_bk[count1:count2,..] = zt_bk[indxobs[count1:count2,..]] ;
ztilda_obs_md[count1:count2,..] = zt_md[indxobs[count1:count2,..]] ;
etilda_obs_bk[count1:count2,..] = et_bk[indxobs[count1:count2,..]] ;
etilda_obs_md[count1:count2,..] = et_md[indxobs[count1:count2,..]] ;

m=m+1;
endo ;

// TASK 2: Pseudo Maximum Likelihood Estimation
// -----------------------------------------------------------------------
zobs = (ztilda_obs_bk[..,1:kvpfc] | zeros(nobs,kvpfc))
~(zeros(nobs,kvpfc) | ztilda_obs_md[..,1:kvpfc])
~(ztilda_obs_bk[..,kvpfc+1:kvpfc+kz] | ztilda_obs_md[..,kvpfc+1:kvpfc+kz]) ;
eobs = etilda_obs_bk | etilda_obs_md ;
{thetaest,varest,likelihood} = miprobit((yobs[..,1] | yobs[..,2]),zobs,eobs,zeros(ktot,1),namesb,1) ;
// TASK 3: Updating Conditional Choice Probabilities
// -----------------------------------------------------------------------
pchoice = cdfn(ztilda_bk*theta_bk + etilda_bk) ~ cdfn(ztilda_md*theta_md + etilda_md) ;
// Checking for Convergence
criterion = maxc(abs(thetaest-theta0))

theta0 = thetaest;
npliter = npliter+1;
endo;

// Observed Conditional Choice Probabilities:
pest_obs = cdfn(ztilda_obs_bk * theta_bk + etilda_obs_bk)
~cdfn(ztilda_obs_md * theta_md + etilda_obs_md);

retp(thetaest,varest,pchoice,pest_obs);
endp;

// PART 4: ESTIMATION OF INITIAL CCPs
prob_freq = zeros(nmarket*nstate,nplayer);
market = 1;
do while market<=nmarket;
count1 = (market-1)*nstate + 1;
count2 = market*nstate;
yyy = a_bk[(market-1)*nyear+1:market*nyear] ~a_md[(market-1)*nyear+1:market*nyear];
xxx = x_bk[(market-1)*nyear+1:market*nyear] ~x_md[(market-1)*nyear+1:market*nyear];
buff = freqprob(yyy,xxx,vstate);
prob_freq[count1:count2,.] = freqprob(yyy,xxx,vstate);
market = market+1;
endo;

// Alternatively, the user could initialize the NPL algorithm using
// a vector of constant probabilities, e.g.,
// prob_freq = (1/2)*ones(nmarket*nstate,nplayer);
// or using random draws from a uniform distribution, e.g.,
// prob_freq = rndu(nmarket*nstate,nplayer);

// PART 5: NPL ESTIMATION
{best,varb,pstate,pobs} = npl_bkmd((a_bk ~ a_md), (x_bk ~ x_md), marketsize, zmarket, prob_freq, vstate, dfact, maxiter, namesb);
end;
References


