Quantitative Economics 6 (2015), 567–597 1759-7331/20150567

Estimating dynamic discrete-choice games of incomplete information

Michael Egesdal
Department of Economics, Harvard University

Zhenyu Lai
Department of Economics, Harvard University

Che-Lin Su
The University of Chicago Booth School of Business

We investigate the estimation of models of dynamic discrete-choice games of incomplete information, formulating the maximum-likelihood estimation exercise as a constrained optimization problem that can be solved using state-of-the-art constrained optimization solvers. Under the assumption that only one equilibrium is played in the data, our approach avoids repeatedly solving the dynamic game or finding all equilibria for each candidate vector of the structural parameters. We conduct Monte Carlo experiments to investigate the numerical performance and finite-sample properties of the constrained optimization approach for computing the maximum-likelihood estimator, the two-step pseudo-maximum-likelihood estimator, and the nested pseudo-likelihood estimator, implemented by both the nested pseudo-likelihood algorithm and a modified nested pseudo-likelihood algorithm.

Keywords. Dynamic discrete-choice games of incomplete information, maximum-likelihood estimator, constrained optimization, nested pseudo-likelihood estimator.

JEL classification. C57, C73.

Michael Egesdal: egesdal@fas.harvard.edu
Zhenyu Lai: zlai@wayfair.com

Che-Lin Su, our dear professor and coauthor, tragically passed away earlier this year. We hope that this paper inspires in other economists the same enthusiasm for computational economics that he instilled in us. We have benefited greatly from discussions with Timothy B. Armstrong, C. Lanier Benkard, Steven T. Berry, Xiaohong Chen, Jean-Pierre Dubé, Philip A. Haile, Christian B. Hansen, Kenneth L. Judd, Ariel Pakes, John P. Rust, and seminar participants at Yale University, University of British Columbia, University of Chicago, Harvard University, 2013 Cowles Summer Conference in Structural Microeconomics, and 2014 Stanford SITE Conference. We are grateful to Harry J. Paarsch, Hiro Kasahara, the co-editor, and anonymous referees for their detailed and helpful comments on this paper. Che-Lin Su acknowledges financial support from the Robert King Steel Faculty Fellowship. All errors are our own.

DOI: 10.3982/QE430

1. Introduction

Empirical models of dynamic games of incomplete information are an important framework within which to study firms' strategic behavior. In the past decade, developing econometric methods to estimate these models has become an active research topic in the empirical industrial organization and applied econometrics literatures. As models of dynamic games become increasingly sophisticated, estimating the underlying structural parameters and decision policies adopted by firms becomes increasingly challenging computationally. For example, Igami (2013, 2014) both estimate a dynamic oligopoly game using a nested fixed-point algorithm that could easily take a few weeks to compute.

The high computational costs of solving dynamic games during the estimation stage has motivated researchers to propose econometric methods that provide consistent estimates in large-sample theory, and that are computationally light and easy to implement in practice. Most of these computationally simple methods belong to the class of two-step estimators. For example, see Bajari, Benkard, and Levin (2007), Pakes, Ostrovsky, and Berry (2007), Pesendorfer and Schmidt-Dengler (2008), Arcidiacono and Miller (2011), and Srisuma (2013) as well as Sanches, Silva, and Srisuma (2013). The potential drawbacks of two-step estimators are that their estimates can have large biases in finite samples because insufficient data exist to obtain precise estimates in the first step, and that researchers might not use an appropriate criterion function in the second step; see the discussion in Pakes, Ostrovsky, and Berry (2007).[1] To address these issues, Aguirregabiria and Mira (2007) have proposed the nested pseudo-likelihood (NPL) estimator and the NPL algorithm to compute the NPL estimator. Using Monte Carlo experiments, Aguirregabiria and Mira demonstrated that the NPL estimator is less biased than the two-step pseudo-maximum-likelihood (2S-PML) estimator.

Pakes, Ostrovsky, and Berry (2007), Pesendorfer and Schmidt-Dengler (2008, 2010), and Su (2014) have shown that the NPL algorithm can frequently fail to converge. Even worse, the NPL algorithm may not provide consistent estimates. Kasahara and Shimotsu (2012) analyzed the convergence properties of the NPL algorithm and suggested modifications in implementing the NPL algorithm to improve its convergence. Using a simplified version of a dynamic game model derived from Aguirregabiria and Mira (2007), they illustrated that their modified NPL (NPL-Λ) algorithm indeed converged and performed well in a Monte Carlo experiment, while the original NPL algorithm failed.

Su and Judd (2012) have proposed a constrained optimization approach to estimating structural models, while Dubé, Fox, and Su (2012) applied the constrained optimization approach to the estimation of random-coefficients logit demand models, successfully solving examples having tens of thousands of variables and constraints. Su (2014) illustrated that the constrained optimization approach can be applied to estimating static games of incomplete information with multiple equilibria (under the assumption that

[1] In their Monte Carlo experiments, Pakes, Ostrovsky, and Berry (2007) find that two-step estimators can be biased in small samples although these biases shrink rapidly as sample size increases. Further, they find that the simplest criterion function performs best. These results are further supported by Pesendorfer and Schmidt-Dengler (2008), who find that biases for two-step estimators are relatively small with sample size $T = 1000$, but not for $T = 100$.

only one equilibrium is played in each market in the data) and that it performed better than the NPL estimator in Monte Carlo experiments. Even so, some researchers remain unsure whether the constrained optimization approach is practical to estimate dynamic games because it requires solving high-dimensional optimization problems, which can be computationally demanding.

Following Su and Judd (2012) as well as Su (2014), we have formulated the maximum-likelihood (ML) estimation problem of dynamic discrete-choice games of incomplete information as a constrained optimization problem. Using the dynamic game provided by the entry/exit model of Aguirregabiria and Mira (2007), we have conducted Monte Carlo experiments to investigate the finite-sample properties and the numerical performance of the 2S-PML estimator, the NPL estimator implemented by the NPL and NPL-Λ algorithms, and the ML estimator implemented by the constrained optimization approach. Our Monte Carlo results suggest that the constrained optimization approach is more robust and reliable than both the NPL and the NPL-Λ algorithms. Indeed, the constrained approach converged for all data sets in all experiments, while the performance of the NPL and NPL-Λ algorithms varied. In some cases, the NPL and NPL-Λ algorithms failed to converge. The constrained optimization approach also works well when the size of the state space in the model increases, but the state transition matrix remains sparse. Although the 2S-PML estimator always converged in our experiments, this estimator is much less accurate than the ML estimator under the constrained optimization approach. Overall, when compared to alternative estimators, using the constrained optimization approach to ML estimation offers valuable returns: reliable convergence, computational speed, and accurate estimates.

We have organized the remainder of the paper as follows: In Section 2, we describe the dynamic model of the entry/exit game as proposed by Aguirregabiria and Mira (2007), while in Section 3, we present the constrained optimization formulation for the ML estimation of these games and discuss alternative likelihood-based estimators and their associated estimation algorithms. In Section 4, we describe the design of our Monte Carlo experiments, present numerical results, and conduct additional robustness checks. We summarize our conclusions in Section 5 and briefly describe potential future work. An Appendix and replication files, which also include equilibrium solutions, are available in supplementary files on the journal website, http://qeconomics.org/supp/430/supplement.pdf and http://qeconomics.org/supp/430/code_and_data.zip.

2. Model

We consider a model of discrete-time, infinite-horizon dynamic games based on the research of Aguirregabiria and Mira (2007). In each period $t = 1, 2, \ldots, \infty$, N potential entrants exist, each indexed by $i \in \mathcal{I} = \{1, \ldots, N\}$; these players operate in a market characterized by size $s^t \in \mathcal{S} = \{s_1, \ldots, s_L\}$. We assume that market size is observed by all players and evolves according to the exogenous stationary transition probability $f_{\mathcal{S}}(s^{t+1}|s^t)$, where $s^t, s^{t+1} \in \mathcal{S}$.

At the beginning of each period t, player i observes a vector of common-knowledge state variables $\mathbf{x}^t$ and private shocks $\boldsymbol{\varepsilon}_i^t$. Players then simultaneously choose whether to be active in the market. Let $a_i^t \in \mathcal{A} = \{0, 1\}$ denote player i's action in period t and let

$\mathbf{a}^t = (a_1^t, \ldots, a_N^t)$ denote the collection of all players' actions. The common-knowledge state variables $\mathbf{x}^t$ consist of market size and all players' actions in the previous period, namely, $\mathbf{x}^t = (s^t, \mathbf{a}^{t-1}) \in \mathcal{X} = \{\mathcal{S} \times \{\times_{i\in\mathcal{I}}\mathcal{A}\}\}$. Each player i also privately observes $\boldsymbol{\varepsilon}_i^t = \{\varepsilon_i^t(a_i^t)\}_{a_i^t\in\mathcal{A}}$, a vector of choice-contingent shocks to per-period payoffs. We assume that $\varepsilon_i^t(a_i^t)$ has a type-I extreme value distribution that is independent and identically distributed across actions and players as well as over time, and that opposing players do not observe the realization of $\boldsymbol{\varepsilon}_i^t$, but know only its probability density function $g(\boldsymbol{\varepsilon}_i^t)$.

The state variables $(\mathbf{x}^t, \boldsymbol{\varepsilon}_i^t)$ evolve after the decisions $\mathbf{a}^t$ have been made, so their evolution is described by the exogenous probability distribution function $p(\mathbf{x}^{t+1}, \boldsymbol{\varepsilon}_i^{t+1}|\mathbf{x}^t, \boldsymbol{\varepsilon}_i^t, \mathbf{a}^t)$. We further impose the *conditional independence* assumption. That is,

$$p\big[\mathbf{x}^{t+1} = \big(s', \mathbf{a}'\big), \boldsymbol{\varepsilon}_i^{t+1}|\mathbf{x}^t = (s, \tilde{\mathbf{a}}), \boldsymbol{\varepsilon}_i^t, \mathbf{a}^t\big] = f_{\mathcal{S}}\big(s'|s\big)\mathbf{1}\big\{\mathbf{a}' = \mathbf{a}^t\big\}g\big(\boldsymbol{\varepsilon}_i^{t+1}\big), \tag{1}$$

where $\mathbf{1}$ is the indicator function.

Denote by $\boldsymbol{\theta}$ the vector of structural parameters and by $\mathbf{a}_{-i}^t = (a_1^t, \ldots, a_{i-1}^t, a_{i+1}^t, \ldots, a_N^t)$ the current actions of all players other than i in period t. We specify player i's per-period payoff function as $\tilde{\Pi}_i(a_i^t, \mathbf{a}_{-i}^t, \mathbf{x}^t, \boldsymbol{\varepsilon}_i^t; \boldsymbol{\theta}) = \Pi_i(a_i^t, \mathbf{a}_{-i}^t, \mathbf{x}^t; \boldsymbol{\theta}) + \varepsilon_i^t(a_i^t)$, which is additively separable in a common-knowledge component and a private shock. Here, the common-knowledge component $\Pi_i(a_i^t, \mathbf{a}_{-i}^t, \mathbf{x}^t; \boldsymbol{\theta})$ depends on the current actions of all players $\mathbf{a}^t$, publicly observed state variables $\mathbf{x}^t$, and $\boldsymbol{\theta}$. Let $\beta \in (0, 1)$ denote the discount factor. Given the current state $(\mathbf{x}^t, \boldsymbol{\varepsilon}_i^t)$, player i chooses a sequence of decisions to maximize the total expected discounted payoff

$$\max_{\{a_i^t, a_i^{t+1}, a_i^{t+2}, \ldots\}} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t}\tilde{\Pi}_i\big(a_i^\tau, \mathbf{a}_{-i}^\tau, \mathbf{x}^\tau, \boldsymbol{\varepsilon}_i^\tau; \boldsymbol{\theta}\big) \;\Big|\; \big(\mathbf{x}^t, \boldsymbol{\varepsilon}_i^t\big)\right],$$

where the expectation is taken over the state evolution $p(\mathbf{x}^{t+1}, \boldsymbol{\varepsilon}_i^{t+1}|\mathbf{x}^t, \boldsymbol{\varepsilon}_i^t, \mathbf{a}^t)$ given in equation (1) and beliefs about how other players choose their actions.

Since state transition is stationary, we adopt Markov perfect equilibrium as the equilibrium concept. Thus, we can drop the time index t. It is also convenient to characterize the equilibrium in terms of the observed state $\mathbf{x}$. Let $P_i(a_i|\mathbf{x})$ be the conditional choice probability of player i choosing action $a_i \in \mathcal{A}$ at state $\mathbf{x}$. Given $P_j(a_j|\mathbf{x})\ \forall j \neq i$, the expected payoff of the common-knowledge component $\Pi_i(a_i, \mathbf{a}_{-i}, \mathbf{x}; \boldsymbol{\theta})$ for player i from choosing action a_i at state $\mathbf{x}$ is

$$\pi_i(a_i|\mathbf{x}, \boldsymbol{\theta}) = \sum_{\mathbf{a}_{-i}\in\mathcal{A}^{N-1}} \left\{\left[\prod_{a_j\in\mathbf{a}_{-i}} P_j(a_j|\mathbf{x})\right]\Pi_i(a_i, \mathbf{a}_{-i}, \mathbf{x}; \boldsymbol{\theta})\right\}. \tag{2}$$

We denote by $V_i(\mathbf{x})$ the expected value function for player i at state $\mathbf{x}$ and define $\mathbf{P} = \{P_i(a_i|\mathbf{x})\}_{a_i\in\mathcal{A}, i\in\mathcal{I}, \mathbf{x}\in\mathcal{X}}$ and $\mathbf{V} = \{V_i(\mathbf{x})\}_{i\in\mathcal{I}, \mathbf{x}\in\mathcal{X}}$. A Markov perfect equilibrium for this game is a tuple $(\mathbf{V}, \mathbf{P})$ that satisfies the following two systems of nonlinear equations.

I. Bellman Optimality. For all $i \in \mathcal{I}$, $\mathbf{x} \in \mathcal{X}$,

$$\begin{aligned} V_i(\mathbf{x}) &= \sum_{a_i\in\mathcal{A}} P_i(a_i|\mathbf{x})\big[\pi_i(a_i|\mathbf{x}, \boldsymbol{\theta}) + e_i^{\mathbf{P}}(a_i, \mathbf{x})\big] + \beta \sum_{\mathbf{x}'\in\mathcal{X}} V_i\big(\mathbf{x}'\big) f_{\mathcal{X}}^{\mathbf{P}}\big(\mathbf{x}'|\mathbf{x}\big) \\ &= \Psi_i^{\mathbf{V}}(\mathbf{x}; \mathbf{V}, \mathbf{P}, \boldsymbol{\theta}). \end{aligned} \tag{3}$$

The first system of nonlinear equations specifies that for each i and $\mathbf{x}$, given conditional choice probabilities of all players $\mathbf{P}$, the expected value function $V_i(\mathbf{x})$ satisfies the Bellman equation. Here, $f_{\mathcal{X}}^{\mathbf{P}}(\mathbf{x}'|\mathbf{x})$ denotes the state transition probability of $\mathbf{x}$, given $\mathbf{P}$. Specifically,

$$f_{\mathcal{X}}^{\mathbf{P}}\big[\mathbf{x}' = \big(s', \mathbf{a}'\big)|\mathbf{x} = (s, \tilde{\mathbf{a}})\big] = \left[\prod_{j=1}^{N} P_j\big(a_j'|\mathbf{x}\big)\right] f_{\mathcal{S}}\big(s'|s\big). \tag{4}$$

Given the assumption that $\varepsilon_i(a_i)$ follows a type-I extreme value distribution with the scale parameter σ, we have[2]

$$e_i^{\mathbf{P}}(a_i, \mathbf{x}) = \text{Euler's constant} - \sigma \log\big[P_i(a_i|\mathbf{x})\big]. \tag{5}$$

II. Conditional Choice Probability Equation. The second system of equations characterizes conditional choice probabilities (CCP) $\mathbf{P}$. First, we define player i's conditional choice-specific expected value function as

$$v_i(a_i|\mathbf{x}) = \pi_i(a_i|\mathbf{x}, \boldsymbol{\theta}) + \beta \sum_{\mathbf{x}' \in \mathcal{X}} V_i\big(\mathbf{x}'\big) f_i^{\mathbf{P}}\big(\mathbf{x}'|\mathbf{x}, a_i\big), \tag{6}$$

where $f_i^{\mathbf{P}}(\mathbf{x}'|\mathbf{x}, a_i)$ denotes the state transition probability conditional on the current state $\mathbf{x}$, player i's action a_i, and his beliefs $\mathbf{P}$ over the conditional choice probabilities of all other players. Specifically,

$$f_i^{\mathbf{P}}\big[\mathbf{x}' = \big(s', \mathbf{a}'\big)|\mathbf{x} = (s, \tilde{\mathbf{a}}), a_i\big] = f_{\mathcal{S}}\big(s'|s\big)\mathbf{1}\big\{a_i' = a_i\big\} \prod_{j \in \mathcal{I} \setminus i} P_j\big(a_j'|\mathbf{x}\big). \tag{7}$$

After the private shocks $\boldsymbol{\varepsilon}_i = [\varepsilon_i(0), \varepsilon_i(1)]$ are observed, player i chooses action $a_i = j$ if and only if

$$j \in \arg\max_{k \in \mathcal{A}} \big\{v_i(a_i = k|\mathbf{x}) + \varepsilon_i(a_i = k)\big\}.$$

The conditional choice probability is then defined as

$$P_i(a_i = j|\mathbf{x}) = \Pr\Big[\boldsymbol{\varepsilon}_i \big| v_i(a_i = j|\mathbf{x}) + \varepsilon_i(a_i = j) > \max_{k \in \mathcal{A} \setminus j} \big\{v_i(a_i = k|\mathbf{x}) + \varepsilon_i(a_i = k)\big\}\Big].$$

The assumption of a type-I extreme value distribution for $\boldsymbol{\varepsilon_i}$ yields the closed-form expression to characterize conditional choice probabilities $\mathbf{P}$,

$$\begin{aligned} P_i(a_i = j|\mathbf{x}) &= \frac{\exp[v_i(a_i = j|\mathbf{x})]}{\sum_{k \in \mathcal{A}} \exp[v_i(a_i = k|\mathbf{x})]} \\ &= \Psi_i^{\mathbf{P}}(a_i = j|\mathbf{x}; \mathbf{V}, \mathbf{P}, \boldsymbol{\theta}) \quad \forall i \in \mathcal{I}, j \in \mathcal{A}, \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{8}$$

[2]For additional details, see the discussion on page 10 of Aguirregabiria and Mira (2007).

where $v_i(a_i = j|\mathbf{x})$ is defined in equation (6). In what follows, we refer to equation (8) as the CCP equation.

This second system of nonlinear equations specifies that across all states, players' conditional choice probabilities are in mutual best response, given that each player's beliefs are consistent with the choice-specific expected value functions of all players.

To simplify the notation, we define $\Psi^{\mathbf{P}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}) = \{\Psi_i^{\mathbf{P}}(a_i = j|\mathbf{x}; \mathbf{V}, \mathbf{P}, \boldsymbol{\theta})\}_{a_i \in \mathcal{A}, i \in \mathcal{I}, \mathbf{x} \in \mathcal{X}}$ and $\Psi^{\mathbf{V}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}) = \{\Psi_i^{\mathbf{V}}(\mathbf{x}; \mathbf{V}, \mathbf{P}, \boldsymbol{\theta})\}_{i \in \mathcal{I}, \mathbf{x} \in \mathcal{X}}$. We denote the two systems of equations (3) and (8) that characterize a Markov perfect equilibrium at a given vector of parameters $\boldsymbol{\theta}$ by

$$\begin{aligned} \mathbf{V} &= \Psi^{\mathbf{V}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}), \\ \mathbf{P} &= \Psi^{\mathbf{P}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}). \end{aligned} \tag{9}$$

In a dynamic game, multiple Markov perfect equilibria can exist. For a given $\boldsymbol{\theta}$, we denote the set of all Markov perfect equilibria that satisfy equation (9) by

$$\mathrm{SOL}(\Psi, \boldsymbol{\theta}) = \left\{ (\mathbf{P}, \mathbf{V}) \;\middle|\; \begin{array}{l} \mathbf{V} = \Psi^{\mathbf{V}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}) \\ \mathbf{P} = \Psi^{\mathbf{P}}(\mathbf{V}, \mathbf{P}, \boldsymbol{\theta}) \end{array} \right\}.$$

3. Estimation

In this section, we first describe the data generating process and then present a constrained optimization approach for ML estimation of this dynamic game. Finally, we discuss other likelihood-based estimators proposed in the literature.

3.1 *Data generating process*

The data consist of observations from M independent markets over T periods. We assume these M markets follow the same exogenous process $f_{\mathcal{S}}(s'|s)$ for the market-size transitions and that players' decisions are independent across these markets. In each market m and time period t, researchers observe the common-knowledge state variables $\bar{\mathbf{x}}^{mt}$ and players' actions $\bar{\mathbf{a}}^{mt} = (\bar{a}_1^{mt}, \ldots, \bar{a}_N^{mt})$. Let $\mathbf{Z} = \{\bar{\mathbf{a}}^{mt}, \bar{\mathbf{x}}^{mt}\}_{m \in \mathcal{M}, t \in \mathcal{T}}$ denote the collection of data observed across markets and time.

Denote by $\boldsymbol{\theta}^0$ the true value of structural parameters in the population. The vector $(\mathbf{V}^0, \mathbf{P}^0)$ contains the corresponding expected value functions and conditional choice probabilities that simultaneously solve equation (9) at $\boldsymbol{\theta}^0$. If multiple equilibria exist at $\boldsymbol{\theta}^0$, then we assume that only one equilibrium is played across all markets in the data, a common assumption in the literature; see Aguirregabiria and Mira (2007), Bajari, Benkard, and Levin (2007), and Pakes, Ostrovsky, and Berry (2007) as well as Pesendorfer and Schmidt-Dengler (2008).[3] Thus, the data $\mathbf{Z} = \{\bar{\mathbf{a}}^{mt}, \bar{\mathbf{x}}^{mt}\}_{m \in \mathcal{M}, t \in \mathcal{T}}$ are generated from *only one* Markov perfect equilibrium $(\mathbf{V}^0, \mathbf{P}^0)$ at the true parameter values $\boldsymbol{\theta}^0$.

[3] Moment inequality estimators, such as those investigated in Ciliberto and Tamer (2009) or Pakes, Porter, Ho, and Ishii (2011) or Tamer (2003), do not require this assumption.

3.2 *ML estimation*

In this subsection, we define the ML estimator and present an equivalent reformulation of the ML estimator as a constrained optimization problem.

For a given vector $\boldsymbol{\theta}$, let $(\mathbf{P}(\boldsymbol{\theta}), \mathbf{V}(\boldsymbol{\theta})) \in \mathrm{SOL}(\Psi, \boldsymbol{\theta})$ be an equilibrium that satisfies equation (9). Given data $\mathbf{Z} = \{\bar{\mathbf{a}}^{mt}, \bar{\mathbf{x}}^{mt}\}_{m\in\mathcal{M},t\in\mathcal{T}}$, the logarithm of the likelihood of observing data $\mathbf{Z}$ at the parameters $\boldsymbol{\theta}$ is

$$L(\mathbf{Z};\boldsymbol{\theta}) = \max_{(\mathbf{P}(\boldsymbol{\theta}),\mathbf{V}(\boldsymbol{\theta}))\in\mathrm{SOL}(\Psi,\boldsymbol{\theta})} \frac{1}{M}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T}\log P_i\bigl(\bar{a}_i^{mt}|\bar{\mathbf{x}}^{mt}\bigr)(\boldsymbol{\theta}).$$

The ML estimator is then defined as

$$\begin{aligned}\boldsymbol{\theta}^{\mathrm{ML}} &= \arg\max_{\boldsymbol{\theta}} L(\mathbf{Z};\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}}\left\{\max_{(\mathbf{P}(\boldsymbol{\theta}),\mathbf{V}(\boldsymbol{\theta}))\in\mathrm{SOL}(\Psi,\boldsymbol{\theta})} \frac{1}{M}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T}\log P_i\bigl(\bar{a}_i^{mt}|\bar{\mathbf{x}}^{mt}\bigr)(\boldsymbol{\theta})\right\}.\end{aligned} \tag{10}$$

This formulation motivates researchers to apply the nested fixed-point algorithm of Rust (1987) to compute a solution of the ML estimator: in the outer loop, choose $\boldsymbol{\theta}$ to maximize the logarithm of the likelihood function $L(\mathbf{Z};\boldsymbol{\theta})$; in the inner loop, for a given $\boldsymbol{\theta}$, find all Markov perfect equilibria $(\mathbf{P}(\boldsymbol{\theta}), \mathbf{V}(\boldsymbol{\theta})) \in \mathrm{SOL}(\Psi, \boldsymbol{\theta})$ to correctly evaluate the objective function $L(\mathbf{Z};\boldsymbol{\theta})$ at $\boldsymbol{\theta}$.

Researchers face two computational challenges when they apply the nested fixed-point algorithm. First, one has to solve for all the Markov perfect equilibria at each candidate of the structural parameter vector when computing a ML estimator; see Aguirregabiria and Mira (2007, p. 16) as well as Kasahara and Shimotsu (2012). Second, the objective function $L(\mathbf{Z};\boldsymbol{\theta})$ can be a discontinuous function in $\boldsymbol{\theta}$; see Su (2014) for such an example in static discrete-choice games. In what follows, we reformulate the ML estimator (10) as a constrained optimization problem that overcomes these two computational challenges: we do not need to solve for all Markov perfect equilibria, and the objective function and constraints in our reformulation are smooth functions.

Suppose that with data $\mathbf{Z} = \{\bar{\mathbf{a}}^{mt}, \bar{\mathbf{x}}^{mt}\}_{m\in\mathcal{M},t\in\mathcal{T}}$, observed actions $\bar{\mathbf{a}}^{mt}$ at observed state $\bar{\mathbf{x}}^{mt}$ are generated by some choice probabilities $\mathbf{P}$ for all m and t. We then define the logarithm of the augmented likelihood function as

$$\mathcal{L}(\mathbf{Z};\mathbf{V},\mathbf{P},\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T}\log\Psi_i^{\mathbf{P}}\bigl(\bar{a}_i^{mt}|\bar{\mathbf{x}}^{mt};\mathbf{V},\mathbf{P},\boldsymbol{\theta}\bigr). \tag{11}$$

To ensure that the conditional choice probabilities $\mathbf{P}$ and expected value functions $\mathbf{V}$ are consistent with a Markov perfect equilibrium at the given structural parameters $\boldsymbol{\theta}$, we impose equation (9) as constraints. Thus, a constrained optimization formulation of the

ML estimation problem of this dynamic game is

$$\begin{aligned} &\max_{(\boldsymbol{\theta},\mathbf{P},\mathbf{V})} \frac{1}{M}\mathcal{L}(\mathbf{Z};\mathbf{V},\mathbf{P},\boldsymbol{\theta}) \\ &\quad \text{subject to} \quad \mathbf{V} = \Psi^{\mathbf{V}}(\mathbf{V},\mathbf{P},\boldsymbol{\theta}), \\ &\qquad\qquad\quad \mathbf{P} = \Psi^{\mathbf{P}}(\mathbf{V},\mathbf{P},\boldsymbol{\theta}). \end{aligned} \tag{12}$$

The equivalence of the two optimization problems (10) and (12) in the optimal objective value and the optimal solution immediately follows from Proposition 1 in Su and Judd (2012). Aitchison and Silvey (1958) have demonstrated that the ML estimator formulated in (12) is consistent as well as asymptotically normal; see also Section 10.3 in Gourieroux and Monfort (1995). We state this result and the proof in Section S1 of the supplement.

It is easy to verify that the objective function and constraints in (12) are smooth functions of $(\boldsymbol{\theta},\mathbf{P},\mathbf{V})$ and, hence, Newton-based optimization methods can be applied. When solving the constrained optimization problem (12), one does not need to solve for all Markov perfect equilibria at every guess of the structural parameters. Two reasons exist: first, modern constrained optimization solvers do not force the constraints to be satisfied during the iteration process; constraints are satisfied (and an equilibrium solved) only when the iterates converge to a (local) solution; second, the constrained optimization approach only needs to find those equilibria together with structural parameters that are local solutions and satisfy the corresponding first-order conditions of the constrained optimization problem (12). Any pair of a vector of structural parameters and a corresponding equilibrium that does not satisfy the first-order conditions of (12) is not a solution to the ML estimation problem. This characterization permits one to eliminate a large set of equilibria and structural parameters that do not need to be solved by the constrained optimization approach.

3.3 *Alternative dynamic games estimators*

Hotz and Miller (1993) proposed a two-step estimation strategy within the context of single-agent dynamic models. The main insight of Hotz and Miller was to estimate the expected value function directly from the data without solving the Bellman equation, hence reducing the computational burden of estimating dynamic models. Subsequently, researchers have generalized this idea to estimate multi-agent dynamic games and have developed various two-step estimators that are computationally light and easy to implement. For example, see Bajari, Benkard, and Levin (2007) and Pakes, Ostrovsky, and Berry (2007) as well as Pesendorfer and Schmidt-Dengler (2008). A potential drawback of two-step estimators is that they can be more biased than the ML estimator in finite samples, particularly when the first-step estimates are imprecise or if a suitable criterion function is not used in the second step; see the discussion in Pakes, Ostrovsky, and Berry (2007).

In an effort to reduce finite-sample bias associated with two-step estimators, Aguirregabiria and Mira (2007) proposed the NPL estimator and the NPL algorithm,

a recursive computational procedure over the 2S-PML estimator, to compute the NPL estimator. While the NPL estimator performed well in their Monte Carlo experiments, convergence of the NPL algorithm can be a problem. Furthermore, the NPL algorithm may converge to the wrong estimates if the data are generated by an equilibrium that is unstable under best response iterations; see Pesendorfer and Schmidt-Dengler (2010) for such an example and Su (2014) on the performance of the NPL algorithm in a static discrete-choice game. Kasahara and Shimotsu (2012) provided theoretical analysis of the convergence of the NPL algorithm and proposed modified NPL algorithms to alleviate the convergence issue of the NPL algorithm.

We describe three approaches to estimating dynamic games: the 2S-PML estimator, the NPL algorithm of Aguirregabiria and Mira (2007), and the NPL-Λ algorithm of Kasahara and Shimotsu (2012) for computing the NPL estimator.[4] We do not discuss other two-step estimators such as those of Bajari, Benkard, and Levin (2007), Pakes, Ostrovsky, and Berry (2007), and Pesendorfer and Schmidt-Dengler (2008). Instead, we focus on comparing the performance of the ML estimator with that of alternative likelihood-based estimators in our Monte Carlo experiments.

3.3.1 *Two-step pseudo-maximum likelihood* In the first step of a two-step estimator, one can nonparametrically estimate the conditional choice probabilities from the observed data $\mathbf{Z}$ using, for example, the frequency estimator. Denote by $\widehat{\mathbf{P}}$ a consistent estimator of the true conditional choice probabilities $\mathbf{P}^0$. This nonparametric estimate $\widehat{\mathbf{P}}$ is then fixed and used to evaluate the right-hand side of the Bellman optimality equation in (9):[5]

$$\mathbf{V} = \Psi^{\mathbf{V}}(\mathbf{V}, \widehat{\mathbf{P}}, \boldsymbol{\theta}).$$

The second step of the 2S-PML estimator involves solving the optimization problem

$$\begin{aligned} &\max_{(\boldsymbol{\theta},\mathbf{V})} \frac{1}{M}\mathcal{L}(\mathbf{Z}; \mathbf{V}, \widehat{\mathbf{P}}, \boldsymbol{\theta}) \\ &\quad \text{subject to} \quad \mathbf{V} = \Psi^{\mathbf{V}}(\mathbf{V}, \widehat{\mathbf{P}}, \boldsymbol{\theta}). \end{aligned} \tag{13}$$

From equation (3), we can see that once $\mathbf{P}$ is fixed at $\widehat{\mathbf{P}}$, the variables $\mathbf{V}$ and $\boldsymbol{\theta}$ are additively separable. Define $\mathbf{V}_i = [V_i(\mathbf{x})]_{\mathbf{x}\in\mathcal{X}} \in \mathbb{R}^{|\mathcal{X}|}$, $\mathbf{F}_{\mathcal{X}}^{\widehat{\mathbf{P}}} = [f_{\mathcal{X}}^{\widehat{\mathbf{P}}}(\mathbf{x}'|\mathbf{x})]_{\mathbf{x},\mathbf{x}'\in\mathcal{X}} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$, $\widehat{\mathbf{P}}_i(a_i) = [\widehat{P}_i(a_i|\mathbf{x})]_{\mathbf{x}\in\mathcal{X}} \in \mathbb{R}^{|\mathcal{X}|}$, $\mathbf{e}_i^{\widehat{\mathbf{P}}}(a_i) = [e_i^{\widehat{\mathbf{P}}}(a_i, \mathbf{x})]_{\mathbf{x}\in\mathcal{X}} \in \mathbb{R}^{|\mathcal{X}|}$, and $\boldsymbol{\pi}_i(a_i, \boldsymbol{\theta}) = [\pi_i(a_i|\mathbf{x}, \boldsymbol{\theta})]_{\mathbf{x}\in\mathcal{X}} \in \mathbb{R}^{|\mathcal{X}|}$. Equation (3) can then be rewritten in matrix notation as

$$\big[\mathbf{I} - \beta\mathbf{F}_{\mathcal{X}}^{\widehat{\mathbf{P}}}\big]\mathbf{V}_i = \sum_{a_i\in\mathcal{A}} \big[\widehat{\mathbf{P}}_i(a_i) \circ \boldsymbol{\pi}_i(a_i, \boldsymbol{\theta})\big] + \sum_{a_i\in\mathcal{A}} \big[\widehat{\mathbf{P}}_i(a_i) \circ \mathbf{e}_i^{\widehat{\mathbf{P}}}(a_i)\big],$$

[4] Kasahara and Shimotsu (2012) also proposed a recursive projection method and a q-NPL algorithm. These two methods are more computationally demanding and we do not consider them in this paper.

[5] This requires researchers to evaluate $f_{\mathcal{X}}^{\mathbf{P}}(\mathbf{x}'|\mathbf{x})$, $e_i^{\mathbf{P}}(a_i, \mathbf{x})$, and $f_i^{\mathbf{P}}(\mathbf{x}'|\mathbf{x}, a_i)$ using $\widehat{\mathbf{P}}$ in equations (4), (5), and (7), respectively.

where $\mathbf{I}$ is an identity matrix in $\mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ and the notation $\mathbf{A}\circ\mathbf{B}$ denotes the Hadamard product of two matrices $\mathbf{A}$ and $\mathbf{B}$. Thus, one can explicitly express $\mathbf{V}_i$ in terms of structural parameters $\boldsymbol{\theta}$,

$$\mathbf{V}_i = \big[\mathbf{I} - \beta \mathbf{F}_{\mathcal{X}}^{\widehat{\mathbf{P}}}\big]^{-1}\left\{\sum_{a_i\in\mathcal{A}}\big[\widehat{\mathbf{P}}_i(a_i)\circ\boldsymbol{\pi}_i(a_i,\boldsymbol{\theta})\big] + \sum_{a_i\in\mathcal{A}}\big[\widehat{\mathbf{P}}_i(a_i)\circ\mathbf{e}_i^{\widehat{\mathbf{P}}}(a_i)\big]\right\} \quad \forall i\in\mathcal{I}, \tag{14}$$

or in a compact notation

$$\mathbf{V} = \boldsymbol{\Gamma}(\boldsymbol{\theta},\widehat{\mathbf{P}}). \tag{15}$$

By replacing the constraint in problem (13) with equation (15), through a simple elimination of variables $\mathbf{V}$, the optimization problem (13) is equivalent to the unconstrained optimization problem

$$\max_{\boldsymbol{\theta}} \frac{1}{M}\mathcal{L}\big(\mathbf{Z};\boldsymbol{\Gamma}(\boldsymbol{\theta},\widehat{\mathbf{P}}),\widehat{\mathbf{P}},\boldsymbol{\theta}\big).$$

The 2S-PML estimator is then defined as

$$\boldsymbol{\theta}^{\text{2S-PML}} = \arg\max_{\boldsymbol{\theta}} \frac{1}{M}\mathcal{L}\big(\mathbf{Z};\boldsymbol{\Gamma}(\boldsymbol{\theta},\widehat{\mathbf{P}}),\widehat{\mathbf{P}},\boldsymbol{\theta}\big). \tag{16}$$

The 2S-PML estimator is considered to be computationally light because it avoids solving the CCP equation in (9); researchers estimate $\widehat{\mathbf{P}}$ directly from the data. Nevertheless, solving the optimization problem (16) or, equivalently, solving problem (13) in the second step, although easier than problem (12) for the ML estimator, may not be trivial. Researchers still need to solve the Bellman equation for each player as constraints in (13) or invert the matrix $[\mathbf{I}-\beta\mathbf{F}_{\mathcal{X}}^{\widehat{\mathbf{P}}}]$ in (14) for every guess of structural parameters $\boldsymbol{\theta}$ in solving the unconstrained optimization problem (16), a task that can be computationally expensive when the size of the state space $|\mathcal{X}|$ is large.

Note, too, that at the solution $\boldsymbol{\theta}^{\text{2S-PML}}$, the first-step estimate $\widehat{\mathbf{P}}$ may not satisfy the CCP equation in (9) and, hence is not a Markov perfect equilibrium. In finite samples, the bias in the first-step estimate $\widehat{\mathbf{P}}$ can potentially lead to large biases in parameter estimates $\boldsymbol{\theta}^{\text{2S-PML}}$ in the second step, particularly when the pseudo-likelihood function is used as the criterion function; see the discussion in Pakes, Ostrovsky, and Berry (2007).

3.3.2 *NPL estimator* Aguirregabiria and Mira (2007) proposed an NPL estimator for estimating dynamic discrete-choice games. Any point $(\tilde{\boldsymbol{\theta}},\widetilde{\mathbf{P}})$ that satisfies the following conditions is called an NPL fixed point:

$$\begin{aligned}\tilde{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \frac{1}{M}\mathcal{L}\big(\mathbf{Z};\boldsymbol{\Gamma}(\boldsymbol{\theta},\widetilde{\mathbf{P}}),\widetilde{\mathbf{P}},\boldsymbol{\theta}\big),\\ \widetilde{\mathbf{P}} &= \Psi^{\mathbf{P}}\big(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}},\widetilde{\mathbf{P}}),\widetilde{\mathbf{P}},\tilde{\boldsymbol{\theta}}\big).\end{aligned} \tag{17}$$

In principle, more than one NPL fixed point can exist. An NPL estimator $(\boldsymbol{\theta}^{\text{NPL}},\mathbf{P}^{\text{NPL}})$ is the NPL fixed point that yields the highest objective value in (17). Note that the NPL

estimator satisfies the CCP equation in (9). Thus, one would expect it to perform better than the 2S-PML estimator in finite samples.

Aguirregabiria and Mira (2007) also proposed a computational procedure referred to as the NPL algorithm to find an NPL fixed point. The NPL algorithm recursively iterates over the 2S-PML estimator and is described as follows. First, choose an initial guess of equilibrium probabilities $\widetilde{\mathbf{P}}_0$. For $K \geq 1$, the NPL algorithm iterates the following steps until convergence or until the maximum number of iterations $\bar{K}$ is reached.

Step 1. Given $\widetilde{\mathbf{P}}_{K-1}$, solve

$$\tilde{\boldsymbol{\theta}}_K = \arg\max_{\boldsymbol{\theta}} \frac{1}{M} \mathcal{L}\big(\mathbf{Z}; \boldsymbol{\Gamma}(\boldsymbol{\theta}, \widetilde{\mathbf{P}}_{K-1}), \widetilde{\mathbf{P}}_{K-1}, \boldsymbol{\theta}\big).$$

Step 2. Given $\tilde{\boldsymbol{\theta}}_K$, update $\widetilde{\mathbf{P}}_K$ by

$$\widetilde{\mathbf{P}}_K = \Psi^{\mathbf{P}}\big(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1}), \widetilde{\mathbf{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\big);$$

increase K by 1.

To declare convergence of the NPL algorithm, the condition

$$\big\|(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_K) - (\tilde{\boldsymbol{\theta}}_{K-1}, \widetilde{\mathbf{P}}_{K-1})\big\| \leq \texttt{tol}_{\text{NPL}}, \tag{18}$$

needs to be satisfied, where $\texttt{tol}_{\text{NPL}}$ is the convergence tolerance and is chosen to be a small number, for example, `1.0e-6`. If the NPL algorithm converges after K iterations with $K \leq \bar{K}$, then $(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1})$ approximately satisfies the NPL fixed-point conditions (17). If the maximum number of iterations $\bar{K}$ is reached before the NPL algorithm converges, then we declare a failed run and restart with a new initial guess.

Researchers have expressed concerns involving the convergence properties of the NPL algorithm. For example, see Pakes, Ostrovsky, and Berry (2007), Pesendorfer and Schmidt-Dengler (2010), and Su (2014). While Aguirregabiria and Mira (2007) reported that they always obtained convergence of the NPL algorithm in their Monte Carlo experiments, examples in Su (2014) indicate that the NPL algorithm often fails to converge, even in static discrete-choice games. Even worse, Pesendorfer and Schmidt-Dengler (2010) demonstrated in a stylized example and Su (2014) demonstrated in Monte Carlo experiments of static discrete-choice games that NPL can converge to the wrong estimates. In a recent paper, Kasahara and Shimotsu (2012) demonstrated that the NPL algorithm will converge, provided that a local stability condition is satisfied at the solution. Without knowing the true parameter values, however, this local stability condition cannot be verified. Therefore, in practice, the theoretical analysis provided by Kasahara and Shimotsu does not inform researchers a priori whether the NPL algorithm will converge or whether it converges to the correct estimates. In summary, while the NPL estimator is well defined, the NPL algorithm may fail to converge; even if the NPL algorithm converges, in some cases, it may fail to recover the true values of the underlying primitives, and using different initial guesses will not help to alleviate this problem.

3.3.3 *A modified NPL algorithm* To improve the convergence properties of the NPL algorithm, Kasahara and Shimotsu (2012) introduced the NPL-Λ algorithm to compute an NPL estimator. The NPL-Λ algorithm alters the updating of $\widehat{\mathbf{P}}_K$ in Step 2 of the NPL algorithm to

$$\widetilde{\mathbf{P}}_K = \big(\Psi^{\mathbf{P}}\big(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1}), \widetilde{\mathbf{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\big)\big)^{\lambda}(\widetilde{\mathbf{P}}_{K-1})^{1-\lambda}, \tag{19}$$

where λ is chosen to be between 0 and 1. Note that when $\lambda = 1$, the NPL-Λ algorithm is identical to the NPL algorithm; when $\lambda = 0$, $\widetilde{\mathbf{P}}_K$ is not updated by the algorithm, and the NPL-Λ algorithm solves the 2S-PML estimator (16).

The idea behind modifying the second step of the NPL algorithm is similar to that of the successive over- and underrelaxation methods in numerical analysis. Such methods are used to improve the contraction rate of diverging iterative processes; see Ortega and Rheinboldt (1970). The scalar λ represents a partial step length used in the dampening procedure. Ideally, the choice of λ depends on the spectral radius of a Jacobian matrix at the solution $\boldsymbol{\theta}^{\text{NPL}}$. In practice, since $\boldsymbol{\theta}^{\text{NPL}}$ is unknown prior to estimation, the choice of the value of λ becomes a delicate issue. If λ is chosen to be close to 1, then the NPL-Λ algorithm may not converge (if the NPL algorithm does not converge); if λ is chosen to be close to 0, then the NPL-Λ algorithm takes small incremental steps and may need many iterations and, hence, much longer computational time to converge. Kasahara and Shimotsu proposed computing the spectral radius of the Jacobian matrix evaluated at the two-step estimator or, if it is computationally demanding to calculate all eigenvalues of the Jacobian matrix, using a small value of λ. However, as discussed earlier, using a small value of λ will lead to more iterations and longer computational time before the NPL-Λ algorithm converges.

The convergence criterion for the NPL-Λ algorithm also warrants some discussion. If researchers simply use criterion (18) to determine the convergence of the NPL-Λ algorithm, there is no guarantee that the CCP equation in (17) will be satisfied with the desired accuracy, $\texttt{tol}_{\text{NPL}}$, when the NPL-$\Lambda$ algorithm converges. In fact, one can show that when the NPL-Λ algorithm declares criterion (18) to be satisfied, the error of the CCP equation in (17) can remain as large as $\frac{\texttt{tol}_{\text{NPL}}}{\lambda}$; that is,

$$\big\|\Psi^{\mathbf{P}}\big(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1}), \widetilde{\mathbf{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\big) - \widetilde{\mathbf{P}}_{K-1}\big\| \leq \frac{\texttt{tol}_{\text{NPL}}}{\lambda}. \tag{20}$$

If researchers choose a very small value for λ (for example, $\lambda = \texttt{1.0e-5}$) but use the convergence criterion (18) with $\texttt{tol}_{\text{NPL}} = \texttt{1.0e-6}$, then the NPL-$\Lambda$ algorithm may declare convergence quickly but the error of the CCP equation in (17) can be as large as $\frac{\texttt{tol}_{\text{NPL}}}{\lambda} = \texttt{1.0e-1}$.

In determining the convergence of the NPL-Λ algorithm, we use the condition

$$\left\| \begin{matrix} (\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_K) - (\tilde{\boldsymbol{\theta}}_{K-1}, \widetilde{\mathbf{P}}_{K-1}) \\ \Psi^{\mathbf{P}}\big(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1}), \widetilde{\mathbf{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\big) - \widetilde{\mathbf{P}}_{K-1} \end{matrix} \right\| \leq \texttt{tol}_{\text{NPL}}. \tag{21}$$

If this condition is met after K iterations with $K \leq \bar{K}$, then we declare the convergence of the NPL-Λ algorithm and use $(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1})$ as a solution to the NPL fixed point (17).

This convergence criterion (21) ensures that when the NPL-Λ algorithm converges, the iterate $(\tilde{\boldsymbol{\theta}}_K, \widetilde{\mathbf{P}}_{K-1})$ approximately satisfies the CCP equation in (17) with an error that is at most $\texttt{tol}_{\text{NPL}}$.

3.4 *Scalability of the constrained optimization approach*

Since the constrained optimization approach requires optimizing over a much larger number of dimensions, one potential concern is its capability to estimate empirically relevant dynamic games; see Aguirregabiria and Nevo (2013) as well as Kasahara and Shimotsu (2012). To address this concern, we demonstrate below that the constraint Jacobian and the Hessian of the Lagrangian of the constrained optimization problem (12) are sparse under certain modeling specifications. State-of-the-art constrained optimization solvers use sparse matrix routines to exploit this sparsity. Consequently, the solvers can accelerate the computations, economize on memory usage, and permit users to solve high-dimensional optimization problems, those on the order of 100,000 variables and constraints.[6]

As discussed in Section 3.3, the size of the optimization problem (13) in the second step of the 2S-PML estimator as well as the NPL and NPL-Λ algorithms is half the size of the constrained optimization problem (12) for ML estimation. For large-scale dynamic games, the problem (13) will be high dimensional as well. Without utilizing sparse matrix techniques, researchers will not be able to solve a high-dimensional problem like (13) or invert the high-dimensional matrix $[\mathbf{I} - \beta \mathbf{F}^{\widehat{\mathbf{P}}}_{\mathcal{X}}]$ in $\mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ in equation (14).

We derive an upper bound on the density of the constraint Jacobian and Hessian of the Lagragian, denoted by D_J and D_H, respectively, of the constrained optimization problem (12).[7] Recall that $|\mathcal{S}|$ is the number of grid points in the market-size state space and $|\theta|$ is the number of structural parameters. Let δ_s denote the maximum incremental change in market size in one period; for example, given s^t, the market size in the next period is $s^{t+1} \in \{s^t - \delta_s, s^t - \delta_s + 1, \ldots, s^t, \ldots, s^t + \delta_s - 1, s^t + \delta_s\}$.

Proposition 1 (Density Bounds).

(a) *Given a binary action space $\mathcal{A} = \{0, 1\}$, we have*

$$D_J \leq \frac{2}{9}\left(\frac{2\delta_s + 1}{N \cdot |\mathcal{S}|} + \frac{|\boldsymbol{\theta}| + 1}{N \cdot |\mathcal{S}| \cdot 2^N} + \frac{1}{|\mathcal{S}| \cdot 2^{N-1}}\right),$$

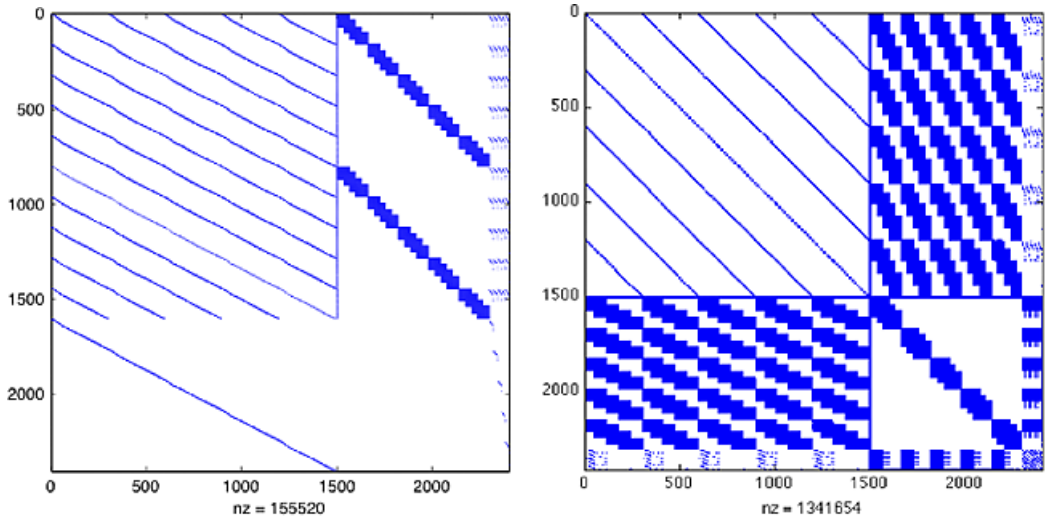
$$D_H \leq \frac{1}{9}\left(\frac{2\delta_s + 1}{N \cdot |\mathcal{S}|} + \frac{4}{|\mathcal{S}| \cdot 2^N} + \frac{4 \cdot (2\delta_s + 1)}{|\mathcal{S}|} + \frac{6 \cdot |\boldsymbol{\theta}|}{N \cdot |\mathcal{S}| \cdot 2^N} + \left(\frac{|\boldsymbol{\theta}|}{N \cdot |\mathcal{S}| \cdot 2^N}\right)^2\right).$$

(b) *The upper bounds on D_J and D_H are decreasing in $|\mathcal{S}|$ and N.*

(c) *For fixed $|\mathcal{S}|$ and N, the upper bounds on D_J and D_H decrease when δ_s decreases.*

[6] All researchers need to do is provide sparsity information to optimization solvers, although insufficient computer memory may continue to limit applications involving more than 100,000 variables and constraints.

[7] The density of a matrix is the ratio between the number of nonzero elements and the total number of elements in the matrix.



(a) Constraint Jacobian (b) Hessian of the Lagrangian function

FIGURE 1. Sparsity pattern of constraint matrices with $|\mathcal{S}| = 5$ and $N = 5$.

The proof is provided in Appendix A. Proposition 1 states that as the size of the state space grows large, the number of nonzero elements in the constraint Jacobian and Hessian matrices grow more slowly than the total number of elements. Even though the total number of nonzero elements still grows, the constraint Jacobian and Hessian matrices become more sparse, which helps to alleviate the increase in computational burden arising from having more variables and constraints. We calculate the density of the constraint Jacobian and Hessian matrices for different values of $|\mathcal{S}|$, N, and $|\theta|$ in Table 14 in Appendix B. Figure 1 illustrates the sparsity pattern of the constraint Jacobian and the Hessian for an example with $N = 5$, $|S| = 5$, and $\delta_s = 1$, which results in a constrained optimization problem with 2400 constraints and 2408 variables. The densities of the corresponding constraint Jacobian and Hessian matrices for this example are around 2.7 percent and 23 percent, respectively.

4. MONTE CARLO EXPERIMENTS

We conducted Monte Carlo experiments to investigate the performance of the ML estimator, the 2S-PML estimator, and the NPL estimator implemented by both the NPL algorithm and the NPL-Λ algorithm. We describe the experimental design in Section 4.1 and report the Monte Carlo results in Section 4.2.

4.1 *Experimental design*

We considered three experiment specifications, with two cases in each experiment. In the first experiment, we used the example of Kasahara and Shimotsu (2012), which is a simplified version of the example of Aguirregabiria and Mira (2007). In the second experiment, we used the example of Aguirregabiria and Mira (2007). In the third experiment,

we increased the set of possible market-size values used in the second experiment. We describe the details of our experimental design below.

EXPERIMENT 1 (Kasahara and Shimotsu (2012) Example). This example has $N = 3$ players. The set of possible values for market size is $\mathcal{S} = \{2, 6, 10\}$ and the total number of grid points in the state space is $|\mathcal{X}| = |\mathcal{S}| \times |\mathcal{A}|^N = 3 \times 2^3 = 24$. The common-knowledge component of the per-period payoff Π_i is given as

$$\Pi_i\big(a_i^t, \mathbf{a}_{-i}^t, \mathbf{x}^t; \boldsymbol{\theta}\big) = \begin{cases} \theta^{\mathrm{RS}} \log\big(s^t\big) - \theta^{\mathrm{RN}} \log\Big(1 + \sum_{j \neq i} a_j^t\Big) \\ \quad - \theta_i^{\mathrm{FC}} - \theta^{\mathrm{EC}}\big(1 - a_i^{t-1}\big), & \text{if } a_i^t = 1, \\ 0, & \text{if } a_i^t = 0, \end{cases}$$

where $\boldsymbol{\theta} = (\theta^{\mathrm{RS}}, \theta^{\mathrm{RN}}, \boldsymbol{\theta}^{\mathrm{FC}}, \theta^{\mathrm{EC}})$ is the vector of structural parameters with $\boldsymbol{\theta}^{\mathrm{FC}} = \{\theta_i^{\mathrm{FC}}\}_{i=1}^N$. For this experiment, the ML estimator solves the constrained optimization problem (12) with 216 constraints and 218 variables.

Following Kasahara and Shimotsu (2012), we chose the discount factor $\beta = 0.96$ and the scale parameter of the type-I extreme value distribution $\sigma = 1$. We fixed the values of structural parameters $\boldsymbol{\theta}^{\mathrm{FC}} = (1.0, 0.9, 0.8)$ and $\theta^{\mathrm{EC}} = 1$, and estimated only θ^{RS} and θ^{RN}.

We considered two sets of parameter values for θ^{RS} and θ^{RN} in this experiment:

$$\text{Case 1:} \quad \big(\theta^{\mathrm{RN}}, \theta^{\mathrm{RS}}\big) = (2, 1);$$

$$\text{Case 2:} \quad \big(\theta^{\mathrm{RN}}, \theta^{\mathrm{RS}}\big) = (4, 1).$$

EXPERIMENT 2 (Aguirregabiria and Mira (2007) Example). This example has $N = 5$ players and five possible values for market size, $\mathcal{S} = \{1, 2, \ldots, 5\}$. The number of points in the state space is $|\mathcal{X}| = |\mathcal{S}| \times |\mathcal{A}|^N = 5 \times 2^5 = 160$. The function Π_i is given as[8]

$$\Pi_i\big(a_i^t, \mathbf{a}_{-i}^t, \mathbf{x}^t; \boldsymbol{\theta}\big) = \begin{cases} \theta^{\mathrm{RS}} s^t - \theta^{\mathrm{RN}} \log\Big(1 + \sum_{j \neq i} a_j^t\Big) \\ \quad - \theta_i^{\mathrm{FC}} - \theta^{\mathrm{EC}}\big(1 - a_i^{t-1}\big), & \text{if } a_i^t = 1, \\ 0, & \text{if } a_i^t = 0. \end{cases}$$

Following Aguirregabiria and Mira (2007), we fixed $\beta = 0.95$ and $\sigma = 1$. In this experiment, we estimated all the structural parameters $\boldsymbol{\theta}$. For this experiment, the ML estimator solves the constrained optimization problem (12) with 2400 constraints and 2408 variables.

We chose $\boldsymbol{\theta}^{\mathrm{FC}} = (1.9, 1.8, 1.7, 1.6, 1.5)$ and $\theta^{\mathrm{EC}} = 1$ as true parameter values. For θ^{RN} and θ^{RS}, we considered two cases:

$$\text{Case 3:} \quad \big(\theta^{\mathrm{RN}}, \theta^{\mathrm{RS}}\big) = (2, 1);$$

$$\text{Case 4:} \quad \big(\theta^{\mathrm{RN}}, \theta^{\mathrm{RS}}\big) = (4, 2).$$

[8]The first term in Π_i is given as $\theta^{\mathrm{RS}} \log(s^t)$ in equation (48) of Aguirregabiria and Mira (2007); however, their Gauss code `am_econometrica_2007_montecarlo.prg` used the term $\theta^{\mathrm{RS}} s^t$. Thus, we decided to follow the specification in their code.

Note that the choices of parameter values in Case 3 are the same as those in Experiment 3 in Aguirregabiria and Mira (2007).

Experiment 3 (Examples With Increasing $|\mathcal{S}|$, the Number of Market-Size Values). In this experiment, we considered two sets of market size values:

Case 5: $|\mathcal{S}| = 10$ with $\mathcal{S} = \{1, 2, \ldots, 10\}$;

Case 6: $|\mathcal{S}| = 15$ with $\mathcal{S} = \{1, 2, \ldots, 15\}$.

All other specifications remain the same as those in Case 3 in Experiment 2. Our purpose is to investigate the performance of these estimators when estimating games with a larger number of states. For this experiment, the ML estimator solves the constrained optimization problem (12) with 4800 constraints and 4808 variables for $|\mathcal{S}| = 10$, and 7200 constraints and 7208 variables for $|\mathcal{S}| = 15$.

In all three experiments, the market-size transition probabilities are given by the $|\mathcal{S}| \times |\mathcal{S}|$ matrix

$$f_{\mathcal{S}}\left(s^{t+1}|s^t\right) = \begin{pmatrix} 0.8 & 0.2 & 0 & \cdots & 0 & 0 \\ 0.2 & 0.6 & 0.2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0.2 & 0.6 & 0.2 \\ 0 & 0 & \cdots & 0 & 0.2 & 0.8 \end{pmatrix}.$$

Finding an equilibrium used in the data generating process Given the model primitives, we solved equation (9) for $(\mathbf{V}, \mathbf{P})$ by using AMPL as the programming language and KNITRO, a nonlinear optimization solver, at the structural parameter values specified in each of the six cases above. For each case, we used 100 different starting values to search for multiple equilibria. The starting values for $\mathbf{P}$ and $\mathbf{V}$ at each state are chosen from equally spaced grid points from 0.1 to 1 and from 0 to 20, respectively. We found only a single equilibrium for each case from those 100 starting values. Hence, we do not consider equilibrium selection issues in the data generating process. The equilibrium solutions for all six cases from our implementation are available in the folder with the replication files.

When we changed the support of market size from $\mathcal{S} = \{1, 2, \ldots, 5\}$ in Case 3 to $\mathcal{S} = \{1, 2, \ldots, 10\}$ and $\mathcal{S} = \{1, 2, \ldots, 15\}$ in Case 5 and 6, respectively, we do not change the nature of equilibria in these models. In fact, further examining the equilibrium solutions for Cases 3, 5, and 6 as reported in the supplement indicate that the equilibrium strategies (or conditional-choice probabilities $P_i(a_i|\mathbf{x})$) are almost identical across these three cases for market size smaller than or equal to 5.

In their Monte Carlo experiments, Aguirregabiria and Mira (2007) and Kasahara and Shimotsu (2012) also stated that they found a unique equilibrium at the true parameter values for the examples in Cases 1 and 3, respectively. Furthermore, the equilibrium we found in those two cases is identical to the equilibrium reported by Kasahara and Shimotsu (2012) and Aguirregabiria and Mira (2007).[9]

[9]We confirmed this finding by checking the output of the Matlab code provided by Kasahara and Shimotsu (2012) for Case 1 and the equilibrium solution reported on Aguirregabiria's website for Case 3.

Data simulation As in Aguirregabiria and Mira (2007) and Kasahara and Shimotsu (2012), we draw the initial observations ($t = 1$) for each market ($m = 1, \ldots, M$) from the steady state distribution implied by the equilibrium computed above. For the subsequent observations ($t = 2, \ldots, T$), we then simulated the observed market size s^t and observed actions $\mathbf{a}^t$ using draws based on market-size transition probabilities f_S and the equilibrium conditional choice probabilities $\mathbf{P}$, respectively.

For each case in Experiments 1 and 2, we constructed data sets of three sizes with $M = 400$ markets and $T = 1$, 10, and 20 periods. For each T, we simulated 100 data sets. For Cases 5 and 6 in Experiment 3, we constructed 50 data sets with $M = 400$ markets and $T = 10$ periods.

Algorithm implementation We used AMPL and KNITRO to solve the constrained optimization problem (12) for the ML estimator and the optimization problem (13) for the second step of the 2S-PML, NPL, and NPL-Λ algorithms.[10] In solving these optimization problems, we used the default relative optimality and feasibility tolerance of `1.0e-6` in the KNITRO solver.

In determining the convergence of the NPL and the NPL-Λ algorithms using criterion (18) and (21), respectively, we chose $\texttt{tol}_{\text{NPL}} = \texttt{1.0e-6}$ as the convergence tolerance. If the difference of parameter values and equilibrium probabilities in successive iterates is less than the chosen tolerance before the maximum number of iterations $\bar{K}$ is reached, then we declared that the NPL or the NPL-Λ algorithm converges; otherwise, we declared that they failed to converge in that run. For Experiment 1, we set the maximum number of NPL and NPL-Λ iterations to be $\bar{K} = 250$. For Experiments 2 and 3, we set $\bar{K}$ to 100. For updating $\widetilde{\mathbf{P}}_K$ in equation (19) in the NPL-Λ algorithm, we chose $\lambda = 0.5$ for all three experiments.

Since the optimization problems (12) and (13) are nonconvex programs, we used multiple starting values to find a better local solution for each estimator. For Experiments 1 and 2, we used 10 starting values for each of the 100 data sets when implementing each estimator; for Experiment 3, we used 5 starting values for each of the 50 data sets. For the starting values of $\boldsymbol{\theta}$, we used 10 equally spaced grid points for Experiments 1 and 2, and 5 equally spaced grid points for Experiment 3, ranging from zero to three times the true parameter values. For the starting values in $\mathbf{P}$ and $\mathbf{V}$, we used the frequency estimator and the mean value of $\mathbf{V}^0$ over all players and states from the equilibrium, respectively.

In reporting summary statistics, the average time per run was computed by taking an average over all runs, including those that failed to converge. However, in computing the mean estimates and standard deviations of the sampling distributions, we ignore data sets that failed to converge.

[10] KNITRO uses sparse matrix techniques to solve the resulting Karush–Kuhn–Tucker system of linear equations of the optimization problems (12) and (13). In our implementation, we use an interior-point method that directly solves the primal-dual KKT matrix at each step; if the direct approach for a certain step cannot be guaranteed to be of good quality, then KNITRO switches to a projected conjugate gradient approach. In our implementation of the 2S-PML, NPL, and NPL-Λ algorithms, we solve the optimization problem (13); we do not directly invert the matrix in (14).

4.2 *Numerical results*

In this subsection, we discuss the results of our Monte Carlo experiments.

EXPERIMENT 1. In Table 1, we have collected the results for Case 1, with $(\theta_{RN}, \theta_{RS}) = (2, 1)$. In this case, all estimation algorithms converged for all data sets. All estimators produced fairly precise estimates, except for the 2S-PML estimator with $T = 1$. As expected, these estimates become more precise as T increases. Recall that for each data set, we used 10 starting values. The constrained optimization approach converged for around 920 runs for $T = 1$ and 980 runs for $T = 20$; all the other algorithms converged in all 1000 runs. Note that the 2S-PML estimator was around 2–25 times faster than the constrained optimization approach. However, the constrained approach was faster than either NPL or NPL-Λ; with $T = 1$, the constrained approach took only 0.27 seconds per run compared to 0.45 seconds per run for NPL, a factor of about 1.6. The speed advantage increases as T increases. With $T = 20$, the constrained approach took only 0.15 seconds per run compared to 1.01 seconds per run for NPL, a factor of more than 6.

TABLE 1. Monte Carlo results for Case 1.

			Estimates		Data Sets	Runs	CPU Time	Avg. NPL(-Λ)
M	T	Estimator	θ_{RN}	θ_{RS}	Converged	Converged	(in Sec.)	Iter.
		Truth	2	1	–	–	–	–
400	1	MLE	1.895	0.961	100	917	0.27	–
			(0.580)	(0.156)				
400	1	2S-PML	1.134	0.753	100	1000	0.02	–
			(0.616)	(0.171)				
400	1	NPL	1.909	0.964	100	1000	0.45	30
			(0.628)	(0.168)				
400	1	NPL-Λ	1.909	0.964	100	1000	0.42	28
		($\lambda = 0.5$)	(0.628)	(0.168)				
400	10	MLE	1.970	0.992	100	964	0.16	–
			(0.158)	(0.042)				
400	10	2S-PML	1.819	0.951	100	1000	0.03	–
			(0.236)	(0.062)				
400	10	NPL	1.963	0.991	100	1000	0.61	22
			(0.191)	(0.050)				
400	10	NPL-Λ	1.963	0.991	100	1000	0.56	20
		($\lambda = 0.5$)	(0.191)	(0.050)				
400	20	MLE	2.001	1.000	100	979	0.15	–
			(0.118)	(0.033)				
400	20	2S-PML	1.923	0.979	100	1000	0.06	–
			(0.158)	(0.042)				
400	20	NPL	1.999	0.999	100	1000	1.01	22
			(0.129)	(0.036)				
400	20	NPL-Λ	1.999	0.999	100	1000	0.91	20
		($\lambda = 0.5$)	(0.129)	(0.036)				

TABLE 2. Monte Carlo results for Case 2.

M	T	Estimator	Estimates θ_{RN}	Estimates θ_{RS}	Data Sets Converged	Runs Converged	CPU Time (in Sec.)	Avg. NPL(Λ) Iter.
		Truth	2	1	–	–	–	–
400	1	MLE	4.055	1.003	100	735	0.61	–
			(0.613)	(0.158)				
400	1	2S-PML	3.107	0.839	100	1000	0.02	–
			(0.442)	(0.099)				
400	1	NPL	N/A	N/A	0	0	4.07	250
			(N/A)	(N/A)				
400	1	NPL-Λ	3.986	0.990	100	1000	1.50	93
		($\lambda = 0.5$)	(0.143)	(0.059)				
400	10	MLE	4.003	1.000	100	767	0.50	–
			(0.039)	(0.016)				
400	10	2S-PML	3.902	0.983	100	1000	0.04	–
			(0.099)	(0.025)				
400	10	NPL	N/A	N/A	0	0	7.61	250
			(N/A)	(N/A)				
400	10	NPL-Λ	4.006	1.001	100	1000	2.34	79
		($\lambda = 0.5$)	(0.047)	(0.018)				
400	20	MLE	4.003	1.001	100	820	0.47	–
			(0.032)	(0.011)				
400	20	2S-PML	3.954	0.992	100	1000	0.06	–
			(0.084)	(0.019)				
400	20	NPL	N/A	N/A	0	0	12.38	250
			(N/A)	(N/A)				
400	20	NPL-Λ	4.002	1.001	100	1000	3.50	72
		($\lambda = 0.5$)	(0.035)	(0.011)				

In Table 2, we have collected the results for Case 2, with $(\theta_{RN}, \theta_{RS}) = (4, 1)$. In this case, the NPL algorithm failed to converge before reaching the maximum number of iterations $\bar{K} = 250$ for all data sets. In contrast, both the constrained approach and 2S-PML converged for all 100 data sets, although the constrained optimization approach converged for only 735 out of 1000 runs for $T = 1$ and 820 runs for $T = 20$. The constrained approach also yielded more precise estimates than the 2S-PML estimator. With $T = 1$, the constrained approach yielded mean estimates of 4.055 for θ^{RN} (standard deviation 0.613) and 1.003 for θ^{RS} (standard deviation 0.158), while 2S-PML gave imprecise mean estimates of 3.107 for θ^{RN} (standard deviation 0.442) and 0.839 for θ^{RS} (standard deviation 0.099).

EXPERIMENT 2. In Tables 3 and 4, we have collected the results of the experiment for Case 3. In this case, NPL converged for only 53 data sets for $T = 1, \ldots, 67$ data sets for $T = 20$.[11] The NPL-Λ algorithm worked quite well in this experiment. It significantly im-

[11] In contrast to our findings, Aguirregabiria and Mira (2007) reported that the NPL algorithm converged for all runs in their experiments for Case 3. The most plausible explanation for this difference in findings

TABLE 3. Convergence results and computational time for Case 3.

M	T	Estimator	Data Sets Converged	Runs Converged	CPU Time (in Sec.)	Avg. NPL(Λ) Iter.
400	1	MLE	100	736	216.31	–
400	1	2S-PML	100	1000	1.34	–
400	1	NPL	53	530	45.85	64
400	1	NPL-Λ ($\lambda = 0.5$)	90	882	36.78	49
400	10	MLE	100	995	32.11	–
400	10	2S-PML	100	1000	1.40	–
400	10	NPL	58	580	52.39	72
400	10	NPL-Λ ($\lambda = 0.5$)	100	1000	24.31	33
400	20	MLE	100	999	29.74	–
400	20	2S-PML	100	1000	1.54	–
400	20	NPL	67	664	55.27	71
400	20	NPL-Λ ($\lambda = 0.5$)	100	1000	23.75	32

TABLE 4. Monte Carlo results on parameter estimates for Case 3.

			Estimates							
M	T	Estimator	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
		True Value	1.9	1.8	1.7	1.6	1.5	1	2	1
400	1	MLE	1.941	1.847	1.765	1.656	1.570	0.959	2.485	1.139
			(0.272)	(0.251)	(0.260)	(0.266)	(0.279)	(0.201)	(1.542)	(0.425)
400	1	2S-PML	1.608	1.496	1.425	1.306	1.196	1.174	0.162	0.433
			(0.222)	(0.213)	(0.214)	(0.210)	(0.187)	(0.141)	(0.295)	(0.093)
400	1	NPL	1.907	1.815	1.716	1.573	1.473	1.074	1.413	0.843
			(0.217)	(0.201)	(0.203)	(0.196)	(0.189)	(0.111)	(0.484)	(0.137)
400	1	NPL-Λ	1.923	1.830	1.740	1.619	1.528	0.997	2.077	1.027
		($\lambda = 0.5$)	(0.241)	(0.231)	(0.235)	(0.237)	(0.238)	(0.145)	(0.994)	(0.282)
400	10	MLE	1.895	1.794	1.697	1.597	1.495	0.990	2.048	1.011
			(0.077)	(0.078)	(0.075)	(0.074)	(0.073)	(0.046)	(0.345)	(0.095)
400	10	2S-PML	1.884	1.774	1.662	1.548	1.425	1.040	0.805	0.671
			(0.066)	(0.069)	(0.065)	(0.062)	(0.057)	(0.039)	(0.251)	(0.068)
400	10	NPL	1.894	1.788	1.688	1.581	1.478	1.010	1.812	0.946
			(0.075)	(0.077)	(0.069)	(0.071)	(0.073)	(0.041)	(0.213)	(0.061)
400	10	NPL-Λ	1.896	1.795	1.697	1.597	1.495	0.991	2.039	1.008
		($\lambda = 0.5$)	(0.077)	(0.079)	(0.076)	(0.074)	(0.073)	(0.044)	(0.330)	(0.091)
400	20	MLE	1.903	1.801	1.701	1.600	1.502	0.996	2.020	1.005
			(0.056)	(0.050)	(0.050)	(0.049)	(0.050)	(0.028)	(0.241)	(0.067)
400	20	2S-PML	1.902	1.795	1.684	1.572	1.459	1.027	1.210	0.785
			(0.052)	(0.046)	(0.042)	(0.042)	(0.043)	(0.025)	(0.198)	(0.052)
400	20	NPL	1.909	1.805	1.704	1.600	1.498	1.006	1.879	0.969
			(0.055)	(0.048)	(0.050)	(0.050)	(0.049)	(0.028)	(0.169)	(0.051)
400	20	NPL-Λ	1.903	1.801	1.701	1.600	1.501	0.996	2.014	1.004
		($\lambda = 0.5$)	(0.055)	(0.050)	(0.049)	(0.048)	(0.050)	(0.029)	(0.250)	(0.069)

proved the convergence properties of the NPL algorithm, converging in 90 data sets for $T = 1$ and all 100 data sets for $T = 10$ and $T = 20$. The NPL-Λ algorithm also obtained more accurate estimates in θ_{RN} and θ_{RS} than those of the NPL algorithm for $T = 20$. The mean estimates of the 2S-PML estimator for parameters θ_{RN} and θ_{RS} are quite biased for $T = 1$ and $T = 10$, and are more than 2 standard deviations away from the true parameter values. The constrained optimization approach converged for all 100 data sets for each T. Its computational time and accuracy of estimates are comparable to those of the NPL-Λ algorithm for $T = 10$ and $T = 20$. For $T = 1$, however, the constrained optimization approach was slow, needing 216 seconds per run, on average; its mean estimates are more biased than those of the NPL-Λ algorithm.

While it might seem surprising that the constrained optimization approach produced worse finite-sample properties than the NPL or NPL-Λ algorithm for $T = 1$, further inspection of the converged likelihood values revealed that the constrained optimization approach yielded higher likelihood values than both the NPL and NPL-Λ algorithms. This indicates that for all 100 data sets with $T = 1$, the constrained optimization approach found a better solution (in terms of the objective value) than the NPL or NPL-Λ algorithm.

In Tables 5 and 6, we have collected the results for Case 4. NPL converged for only 2 out of 100 data sets for $T = 1$ and failed to converge for all 100 data sets for $T = 10$ and $T = 20$. NPL-Λ performed reasonably well in this case: It performed better than the NPL algorithm, but failed more frequently than it did in Case 3, converging in 84 out of 100 data sets for $T = 1$ and only in 53 data sets for $T = 20$. Similar to the findings in Case 3, the 2S-PML estimator produced inaccurate estimates of parameters θ_{RN} and θ_{RS}, particularly for $T = 1$ and $T = 10$; for instance, with $T = 1$, the mean 2S-PML estimates of θ_{RN} and θ_{RS} are 0.624 (standard deviation 0.393) and 0.759 (standard deviation 0.150),

TABLE 5. Convergence results and computational time for Case 4.

M	T	Estimator	Data Sets Converged	Runs Converged	CPU Time (in Sec.)	Avg. NPL(-Λ) Iter.
400	1	MLE	100	582	273.88	–
400	1	2S-PML	100	1000	1.71	–
400	1	NPL	2	20	103.77	99
400	1	NPL-Λ ($\lambda = 0.5$)	84	840	74.22	70
400	10	MLE	100	812	149.65	–
400	10	2S-PML	100	1000	1.59	–
400	10	NPL	0	0	102.69	100
400	10	NPL-Λ ($\lambda = 0.5$)	63	630	78.12	77
400	20	MLE	100	871	121.71	–
400	20	2S-PML	100	1000	1.67	–
400	20	NPL	0	0	107.07	100
400	20	NPL-Λ ($\lambda = 0.5$)	53	530	84.30	79

is probably due to the choice of convergence tolerance; they used `1.0e−5` as the convergence tolerance in their implementation, while we used `1.0e−6`.

TABLE 6. Monte Carlo results on parameter estimates for Case 4.

			Estimates							
M	T	Estimator	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
		True Value	1.9	1.8	1.7	1.6	1.5	1	4	2
400	1	MLE	1.923	1.830	1.723	1.613	1.508	1.023	3.898	1.974
			(0.267)	(0.265)	(0.252)	(0.245)	(0.246)	(0.140)	(0.680)	(0.246)
400	1	2S-PML	1.681	1.595	1.474	1.319	1.073	1.369	0.624	0.759
			(0.255)	(0.241)	(0.241)	(0.227)	(0.208)	(0.144)	(0.393)	(0.150)
400	1	NPL	1.997	1.891	1.747	1.676	1.389	1.481	1.958	1.340
			(0.115)	(0.175)	(0.230)	(0.129)	(0.134)	(0.069)	(0.142)	(0.013)
400	1	NPL-Λ	1.963	1.863	1.759	1.631	1.506	1.056	3.680	1.907
		($\lambda = 0.5$)	(0.273)	(0.272)	(0.255)	(0.258)	(0.263)	(0.147)	(0.739)	(0.269)
400	10	MLE	1.897	1.797	1.697	1.594	1.496	0.993	4.015	2.004
			(0.084)	(0.084)	(0.082)	(0.085)	(0.095)	(0.045)	(0.216)	(0.086)
400	10	2S-PML	1.934	1.824	1.703	1.556	1.338	1.123	2.297	1.409
			(0.090)	(0.085)	(0.079)	(0.079)	(0.085)	(0.049)	(0.330)	(0.117)
400	10	NPL	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
			(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)
400	10	NPL-Λ	1.900	1.801	1.700	1.600	1.500	0.991	4.023	2.007
		($\lambda = 0.5$)	(0.079)	(0.081)	(0.077)	(0.080)	(0.091)	(0.052)	(0.255)	(0.098)
400	20	MLE	1.908	1.806	1.707	1.607	1.514	0.991	4.046	2.017
			(0.057)	(0.056)	(0.053)	(0.055)	(0.059)	(0.031)	(0.137)	(0.054)
400	20	2S-PML	1.946	1.840	1.722	1.593	1.413	1.070	2.931	1.635
			(0.066)	(0.062)	(0.059)	(0.059)	(0.059)	(0.039)	(0.224)	(0.079)
400	20	NPL	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
			(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)
400	20	NPL-Λ	1.905	1.804	1.706	1.607	1.517	0.988	4.077	2.027
		($\lambda = 0.5$)	(0.063)	(0.062)	(0.058)	(0.058)	(0.063)	(0.038)	(0.173)	(0.065)

respectively, while the true values are $\theta_{RN} = 4$ and $\theta_{RS} = 2$. For $T = 20$, the mean estimates of θ_{RN} and θ_{RS} of 2S-PML are 4 standard deviations away from the true values. The constrained optimization approach converged for all 100 data sets for different T, although it converged for only 582 runs (out of 1000) for $T = 1$; it also produced fairly accurate estimates of all structural parameters.

Note that in Experiments 1 and 2, the constrained optimization approach failed to converge more frequently and needed longer computing time for $T = 1$ than for $T = 10$ or 20. One possible explanation for this finding is that with fewer observations in the data ($T = 1$), the likelihood function is flatter than that with more observations ($T = 10$ or 20), which makes the optimization problem in the former case more challenging to solve.

EXPERIMENT 3. In Tables 7 and 8, we have collected the results of Cases 5 and 6. With $|\mathcal{S}| = 10$, the NPL algorithm often failed to converge, finding a solution for only 23 of 50 data sets (or 76 out of 250 runs), and produced highly biased estimates of the parameter θ_{RS} for the converged data sets, with a mean estimate of 1.966 (standard deviation

TABLE 7. Convergence results and computational time for Cases 5 and 6.

$\|S\|$	Estimator	Data Sets Converged	Runs Converged	CPU Time (in Sec.)	Avg. NPL($\text{-}\Lambda$) Iter.
10	MLE	50	240	231.52	–
10	2S-PML	50	250	7.44	–
10	NPL	23	76	552.85	89
10	NPL-Λ ($\lambda = 0.5$)	50	241	289.39	43
15	MLE	50	222	762.78	–
15	2S-PML	50	242	31.45	–
15	NPL	0	0	1560.08	100
15	NPL-Λ ($\lambda = 0.5$)	1	3	1658.08	100

TABLE 8. Monte Carlo results on parameter estimates for Cases 5 and 6.

		Estimates							
$\|S\|$	Estimator	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
	True Value	1.9	1.8	1.7	1.6	1.5	1	2	1
10	MLE	1.882	1.780	1.677	1.584	1.472	0.999	2.031	1.004
		(0.092)	(0.087)	(0.079)	(0.084)	(0.068)	(0.046)	(0.201)	(0.048)
10	2S-PML	1.884	1.792	1.679	1.583	1.469	1.039	1.065	0.755
		(0.102)	(0.088)	(0.082)	(0.087)	(0.068)	(0.048)	(0.222)	(0.053)
10	NPL	1.919	1.810	1.699	1.606	1.485	1.011	1.851	1.966
		(0.092)	(0.089)	(0.068)	(0.079)	(0.071)	(0.050)	(0.136)	(0.036)
10	NPL-Λ	1.884	1.781	1.678	1.584	1.472	0.997	2.032	1.005
	($\lambda = 0.5$)	(0.095)	(0.089)	(0.081)	(0.085)	(0.070)	(0.049)	(0.211)	(0.051)
15	MLE	1.897	1.800	1.694	1.597	1.492	0.983	2.040	1.011
		(0.098)	(0.107)	(0.087)	(0.093)	(0.090)	(0.059)	(0.311)	(0.069)
15	2S-PML	1.792	1.705	1.595	1.506	1.394	1.046	0.766	0.664
		(0.119)	(0.123)	(0.119)	(0.114)	(0.114)	(0.059)	(0.220)	(0.053)
15	NPL	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)
15	NPL-Λ	1.922	1.821	1.671	1.611	1.531	1.012	1.992	1.007
	($\lambda = 0.5$)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)

0.036) versus the true value $\theta_{RS} = 1$. The constrained optimization approach and the NPL-Λ algorithm converged for all 50 data sets and produced similar estimates for all structural parameters. However, with $|\mathcal{S}| = 15$ as shown in Table 6, the NPL algorithm failed to converge for all 50 data sets, and the NPL-Λ algorithm converged for only 1 of 50 data sets.[12] The constrained optimization approach converged for all 50 data sets (or

[12]This observation holds when we loosen the convergence tolerance from `1.0e−6` to `1.0e−5` or increase the number of iterations. The stopping error of the NPL or NPL-Λ algorithm typically cycles back to a large value every few iterations while not falling below `1.0e−5`. Loosening the convergence tolerance even further to `1.0e−4` or `1.0e−3` can induce NPL and NPL-Λ to report convergence more frequently in this experiment.

222 out of 250 runs), and produced accurate estimates of all parameters. In both Cases 5 and 6, the 2S-PML estimator produced highly biased estimates of θ_{RN} and θ_{RS}. In terms of computational speed, the 2S-PML estimator was around 25–30 times faster than the constrained optimization approach in this experiment.

Recall that the specifications in Experiment 3 are identical to those in Case 2 in Experiment 2 except that we increased the number of grid points in the market-size state space from 5 to 10 and 15 in Cases 5 and 6, respectively. It is surprising to see that the NPL and NPL-Λ algorithms failed to converge for almost all data sets when we simply increased the size of the state space to 15, but fixed the true parameter values in the data generating process.

4.3 *Implementation improvements and robustness checks*

In this subsection, we modify our implementation of the constrained optimization approach for the ML estimator and the NPL-Λ algorithm for the NPL estimator in some of the Monte Carlo experiments to investigate the potential for performance improvements. For the ML estimator, we use the 2S-PML estimates as starting values for solving the constrained optimization problem (12); for the NPL-Λ algorithm, we use different values of λ in updating the conditional choice probabilities. We provide more details on our implementation and discuss the numerical results below.

For the ML estimator, instead of using the starting values specified in Section 4.1, we use the 2S-PML estimates as starting values for the constrained optimization problem (12) and rerun the estimation for Experiments 2 and 3. We report the corresponding results for Cases 3 and 4 in Tables 9 and 10, respectively, and the results for Cases 5 and 6 in Table 11.

For Case 3, the average speed of the constrained optimization approach improves from 216 seconds to 42 seconds for $T = 1$, from 32 seconds to 25 seconds for $T = 10$, and from 29 seconds to 23 seconds for $T = 20$; see Tables 3 and 9. For Case 4, the average time per run improves from 273 seconds to 42 seconds for $T = 1$, from 149 seconds to 29 seconds for $T = 10$, and from 121 seconds to 27 seconds for $T = 20$; see Tables 5 and 10. We also observed similar speed improvements with the data sets in Cases 5 and 6 in Experiment 3. For Cases 5 and 6, when using the 2S-PML estimates as starting val-

TABLE 9. Additional ML estimator results using 2S-PML as starting values for Case 3.

		Estimates								Data Sets	CPU Time
M	T	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}	Conv.	(in Sec.)
	Truth	1.9	1.8	1.7	1.6	1.5	1	2	1		
400	1	1.949	1.849	1.764	1.651	1.563	0.983	2.257	1.086	99	42.35
		(0.254)	(0.236)	(0.241)	(0.247)	(0.250)	(0.150)	(1.086)	(0.310)		
400	10	1.895	1.794	1.697	1.597	1.495	0.990	2.048	1.011	100	25.05
		(0.077)	(0.078)	(0.075)	(0.074)	(0.073)	(0.046)	(0.345)	(0.095)		
400	20	1.903	1.801	1.701	1.600	1.502	0.996	2.020	1.005	100	23.61
		(0.056)	(0.050)	(0.050)	(0.049)	(0.050)	(0.028)	(0.241)	(0.067)		

TABLE 10. Additional MLE results using 2S-PML as starting values for Case 4.

		Estimates								Data Sets Conv.	CPU Time (in Sec.)
M	T	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}		
	Truth	1.9	1.8	1.7	1.6	1.5	1	4	2		
400	1	1.947	1.845	1.741	1.632	1.538	1.006	3.989	2.011	100	42.19
		(0.310)	(0.291)	(0.282)	(0.287)	(0.316)	(0.181)	(0.906)	(0.343)		
400	10	1.897	1.797	1.697	1.594	1.496	0.993	4.015	2.004	100	29.19
		(0.084)	(0.084)	(0.082)	(0.085)	(0.095)	(0.045)	(0.216)	(0.086)		
400	20	1.908	1.806	1.707	1.607	1.514	0.991	4.046	2.017	100	27.43
		(0.057)	(0.056)	(0.053)	(0.055)	(0.059)	(0.031)	(0.137)	(0.054)		

TABLE 11. MLE results using 2S-PML estimator as starting values for Cases 5 and 6.

	Estimates								Data Sets Conv.	CPU Time (in Sec.)
$\|S\|$	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}		
Truth	1.9	1.8	1.7	1.6	1.5	1	2	1		
10	1.882	1.780	1.677	1.584	1.472	0.999	2.031	1.004	50	91.41
	(0.092)	(0.087)	(0.079)	(0.084)	(0.068)	(0.046)	(0.201)	(0.048)		
15	1.899	1.803	1.697	1.600	1.494	0.983	2.034	1.010	49	449.06
	(0.098)	(0.106)	(0.085)	(0.093)	(0.090)	(0.059)	(0.304)	(0.067)		

ues for the constrained optimization approach, the average time per run improves from 231 seconds to 91 seconds and from 762 seconds to 449 seconds, respectively; see Tables 7 and 11. Clearly, using the 2S-PML estimates as starting values may decrease average computational time of the ML estimator by as much as a factor of 6.

In terms of performance on the number of data sets converged and the accuracy of the estimates, solving the constrained optimization problem (12) with 2S-PML estimates as starting values produces comparable results to those reported in Tables 3–8. This observation suggests that with a better choice of starting values, the performance of the ML estimator can be improved significantly in terms of decreasing computational time while obtaining estimates that are qualitatively similar.[13]

For the NPL-Λ algorithm, we perform robustness checks by using different values of λ in the updating of $\widetilde{\mathbf{P}}_K$ in equation (19). Recall that the choice of λ affects the contraction rate of the iterative process and researchers may seek to adjust λ to improve convergence, for example, by using a small value of λ as suggested by Kasahara and Shimotsu (2012). However, the choice of the value of λ also affects computational time and the number of NPL-Λ iterations required to converge. When researchers are more conservative and choose the value of λ to be small, the NPL-Λ algorithm may also require more iterations to achieve convergence, which increases computational time. In examining the impact of different λ values on the performance of the NPL-Λ algorithm, we

[13]In practice, researchers should still use multiple starting values, perhaps by perturbing the 2S-PML estimates, in an attempt to find a global solution.

TABLE 12. Additional NPL-Λ results with different λ values for Case 4.

		Estimates								Data Sets Conv.	CPU Time (in Sec.)	NPL-Λ Iter.
T	λ	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}			
	Truth	1.9	1.8	1.7	1.6	1.5	1	4	2			
1	0.9	2.009	1.869	1.743	1.571	1.339	1.301	2.234	1.414	8	78.38	95.6
		(0.266)	(0.282)	(0.285)	(0.311)	(0.275)	(0.119)	(0.222)	(0.107)			
1	0.7	1.970	1.873	1.741	1.612	1.460	1.111	3.349	1.790	54	61.89	71.1
		(0.238)	(0.241)	(0.210)	(0.201)	(0.170)	(0.129)	(0.584)	(0.185)			
1	0.3	2.006	1.916	1.797	1.619	1.409	1.167	2.819	1.621	25	84.27	96.6
		(0.277)	(0.298)	(0.279)	(0.287)	(0.265)	(0.151)	(0.507)	(0.192)			
1	0.1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	87.83	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
10	0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	88.53	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
10	0.7	1.879	1.782	1.678	1.571	1.454	1.016	3.876	1.949	33	76.30	89.7
		(0.081)	(0.081)	(0.077)	(0.073)	(0.076)	(0.047)	(0.216)	(0.083)			
10	0.3	1.873	1.786	1.683	1.560	1.407	1.058	3.581	1.845	11	83.84	99.4
		(0.110)	(0.098)	(0.107)	(0.102)	(0.102)	(0.049)	(0.181)	(0.085)			
10	0.1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	88.03	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
20	0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	92.59	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
20	0.7	1.896	1.787	1.697	1.591	1.485	1.016	3.935	1.972	22	84.54	94.5
		(0.084)	(0.084)	(0.082)	(0.085)	(0.095)	(0.045)	(0.216)	(0.086)			
20	0.3	1.932	1.834	1.731	1.623	1.513	1.016	3.884	1.969	15	85.49	99.4
		(0.068)	(0.066)	(0.068)	(0.065)	(0.069)	(0.026)	(0.133)	(0.053)			
20	0.1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	92.67	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			

set $\lambda = \{0.1, 0.3, 0.7, 0.9\}$ and re-solve the estimation problem for each data set in Cases 4 and 6, the two cases for which the NPL algorithm failed to converge most often.

In Table 12, we report the results of the NPL-Λ algorithm using different λ values for the data sets used in Case 4. Recall that for Case 4, the NPL algorithm converged for only 2 data sets for $T = 1$ and failed to converge for all 100 data sets for $T = 10$ and 20, and the NPL-Λ algorithm with $\lambda = 0.5$ converged in 84 data sets for $T = 1$, 63 data sets for $T = 10$, and 53 data sets for $T = 20$; see Table 5. With $\lambda = 0.9$, we expect the performance of the NPL and the NPL-Λ algorithms to be similar since each iteration of the NPL-Λ algorithm should be similar to that of the NPL algorithm. Indeed, we observe that the NPL-Λ algorithm only converges for 8 data sets for $T = 1$, and fails to converge for all data sets for $T = 10$ and 20.

With $\lambda = 0.1$, the NPL-Λ algorithm takes small steps in updating the conditional choice probabilities. Although, in theory, using a small value of λ may help the NPL-Λ algorithm to converge, the downside is that the iterations progress very slowly and the NPL-Λ algorithm may need many iterations before convergence is achieved. As can be seen in Table 12, the NPL-Λ algorithm fails to converge in all data sets for $T = 1$, 10,

TABLE 13. Additional NPL-Λ results with different λ values for Case 6 ($|S| = 15$).

		Estimates								Data Sets Conv.	CPU Time (in Sec.)	NPL-Λ Iter.
T	λ	$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}			
	Truth	1.9	1.8	1.7	1.6	1.5	1	4	2			
10	0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	1706.26	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
10	0.7	1.922	1.821	1.671	1.611	1.531	1.012	1.992	1.007	1	1679.52	99.6
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
10	0.3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	1766.75	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			
10	0.1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	1764.13	100
		(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)			

and 20. Further inspection of the iteration output indicates that the maximum number of iterations $\bar{K} = 100$ is reached before the convergence condition (21) is satisfied. This observation confirms our conjecture that with a small value of λ, the NPL-Λ algorithm will require more iterations to achieve convergence.

We also report the results of the NPL-Λ algorithm with $\lambda = 0.3$ and 0.7. As expected, the performance of the NPL-Λ algorithm for these two values is better than that of $\lambda = 0.1$ or 0.9. Still, the best convergence rate of the NPL-Λ algorithm is obtained in our original implementation with $\lambda = 0.5$.[14]

In Table 13, we report the results of the NPL-Λ algorithm with different λ values ($\lambda = \{0.1, 0.3, 0.7, 0.9\}$) for the data sets in Case 6. Recall that for Case 6, the NPL algorithm failed to converge in all 50 data sets and the NPL-Λ algorithm with $\lambda = 0.5$ converged in 1 out of 50 data sets; see Table 7. As shown in Table 13, across multiple values of λ, the NPL-Λ algorithm converged for only 1 data set (with $\lambda = 0.7$). In this case, using different values of λ does not help to alleviate the convergence issue of the NPL or the NPL-Λ algorithm.[15]

In summary, choosing an appropriate value for λ in implementing the NPL-Λ algorithm is a delicate issue. As shown in our numerical results, choosing λ to be close to 1 (e.g., $\lambda = 0.9$) does not alter the iteration process of the NPL algorithm too much and, hence, may result in limited improvement in the convergence of the NPL-Λ algorithm. On the other hand, choosing λ to be small (e.g., $\lambda = 0.1$ or 0.01) may, in theory, help the NPL-Λ algorithm achieve convergence, but will require more iterations and longer computational time. The results of our numerical experiments reported in Tables 12 and 13 document and illustrate this trade-off between using different values of λ in implementing the NPL-Λ algorithm.

[14]We also increased $\bar{K}$ to 500 and reran the NPL-Λ algorithm using a few sample data sets for which the NPL-Λ algorithm failed to converge with $\bar{K} = 100$. For $\lambda = 0.1$ and $\lambda = 0.3$, the stopping error bounced between values from roughly `1.0e-4` to `1.0` with no sign of a trend toward convergence. For $\lambda = 0.7$ and $\lambda = 0.9$, the stopping error sometimes went as low as on the order of `1.0e-5`, but then bounced back to values on the order of `1.0e-1` or even larger than `1.0` in a few iterations.

[15]However, using a loose convergence tolerance such as `1.0e-4` or `1.0e-3` may help the NPL-Λ algorithm to report convergence more frequently.

5. Conclusion

In this paper, we have formulated the ML estimation of dynamic discrete-choice games of incomplete information as a constrained optimization problem. We have compared the numerical performance of our constrained approach to the 2S-PML estimator, which suffers from large finite-sample biases in many cases, and to the NPL and NPL-Λ algorithms, which suffer from convergence issues. Our Monte Carlo experiments have demonstrated that the lack of convergence of NPL and NPL-Λ is not a trivial issue in practice. In contrast, the constrained optimization approach for ML estimation produces accurate parameter estimates and has better convergence properties when compared to the NPL and NPL-Λ algorithms. The performance of the constrained optimization approach is robust to changes in both the size of the model and the true values of the structural parameters. Our results suggest that the constrained optimization approach for ML estimation is, indeed, practical and computationally feasible for estimating dynamic games with a moderate number of grid points in the state space as long as the constraint Jacobian and the Hessian of the constrained optimization problem are relatively sparse.

For future research, we plan to explore using the constrained approach in games with unobserved heterogeneity, such as the model in Arcidiacono and Miller (2011). We also plan to improve the numerical implementation of the constrained optimization approach to estimate dynamic games with higher dimensional state spaces.

Appendix A: Proof of Proposition 1

We first consider the constraint Jacobian matrix. For any Bellman optimality constraint gradient row, there are at most $2^N(2\delta_s+1)+|\boldsymbol{\theta}|+2N$ nonzero elements. The first term comes from the derivative with respect to $V_i(\mathbf{x}')$ for the same player i, the second term comes from the derivative with respect to the structural parameters, and the third term comes from the derivative with respect to choice probabilities of all players across the binary action space at the current state $\mathbf{x}$. This applies to $N\cdot|\mathcal{S}|\cdot 2^N$ Bellman optimality constraints. Similarly, inspecting the CCP equation (8) leads, by the same derivation, to at most $|2|^N(2\delta_s+1)+|\boldsymbol{\theta}|+2N$ nonzero elements for each constraint gradient row. This refers nontrivially to each of $N\cdot|\mathcal{S}|\cdot 2^N$ rows. Finally, we add at most two nonzero elements per row for each of the remaining $N\cdot|\mathcal{S}|\cdot 2^N$ CCP equations, forcing the choice probabilities to sum to 1. This corresponds to the probabilities of each player's actions at the current state $\mathbf{x}$.

Summing up these terms yields the numerator as the number of nonzero elements in the constraint Jacobian matrix. The denominator comes from there being $(2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N$ constraints and $(2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N+|\boldsymbol{\theta}|$ variables. This leads to the upper bound

$$\begin{aligned} D_J &\le \frac{N\cdot|\mathcal{S}|\cdot 2^N\cdot(2\cdot(2^N(2\delta_s+1)+|\boldsymbol{\theta}|+2N)+2)}{((2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N)\cdot((2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N+|\boldsymbol{\theta}|)} \\ &= \frac{2}{3}\left(\frac{2^N(2\delta_s+1)+|\boldsymbol{\theta}|+2N+1}{3N\cdot|\mathcal{S}|\cdot 2^N+|\boldsymbol{\theta}|}\right) \\ &\le \frac{2}{9}\left(\frac{2\delta_s+1}{N\cdot|\mathcal{S}|}+\frac{|\boldsymbol{\theta}|+1}{N\cdot|\mathcal{S}|\cdot 2^N}+\frac{1}{|\mathcal{S}|\cdot 2^{N-1}}\right). \end{aligned} \tag{22}$$

The derivation for D_H is similar. We can sum over five terms to construct the numerator of the upper bound. The first corresponds to the $\partial^2\mathbf{V}$ derivative terms, the second corresponds to the $\partial^2\mathbf{P}$ derivative terms, the third corresponds to the $\partial\mathbf{V}\partial\mathbf{P}$ derivative terms, the fourth corresponds to both the $\partial\mathbf{V}\partial\boldsymbol{\theta}$ and $\partial\mathbf{P}\partial\boldsymbol{\theta}$ derivative terms, and the fifth corresponds to the $\partial^2\boldsymbol{\theta}$ derivative terms. The denominator corresponds to the square of the number of variables. Combining these numerator and denominator terms leads to the expression

$$\begin{aligned} D_H \leq{}& \big(N\cdot|\mathcal{S}|\cdot 2^N\cdot\big(2^N\cdot(2\delta_s+1)\big)+|\mathcal{S}|\cdot 2^N\cdot(2N)^2 \\ &+4N^2\cdot|\mathcal{S}|\cdot 2^N\cdot\big(2^N\cdot(2\delta_s+1)\big)+2\cdot\big((2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N\big)\cdot|\boldsymbol{\theta}|+|\boldsymbol{\theta}|^2\big) \\ &/\big((2+1)\cdot N\cdot|\mathcal{S}|\cdot 2^N+|\boldsymbol{\theta}|\big)^2 \\ \leq{}& \frac{1}{9}\left(\frac{2\delta_s+1}{N\cdot|\mathcal{S}|}+\frac{4}{|\mathcal{S}|\cdot 2^N}+\frac{4\cdot(2\delta_s+1)}{|\mathcal{S}|}+\frac{6\cdot|\boldsymbol{\theta}|}{N\cdot|\mathcal{S}|\cdot 2^N}+\left(\frac{|\boldsymbol{\theta}|}{N\cdot|\mathcal{S}|\cdot 2^N}\right)^2\right). \end{aligned} \tag{23}$$

Proposition 1(b) and (c) follow immediately.

Appendix B: Sparsity information for the constraint Jacobian and Hessian matrices

Table 14. Upper bounds on the density of constraint matrices varying $|\mathcal{S}|$ and N, with $\delta_s = 1$.

			Constrained Jacobian		Hessian	
$\lvert\mathcal{S}\rvert$	N	$\boldsymbol{\theta}$	Nonzero Elements	Density D_J	Nonzero Elements	Density D_H
5	5	8	187,200	3.24%	1,676,900	28.9%
6	5	8	224,640	2.70%	2,012,260	24.1%
7	5	8	262,080	2.31%	2,347,620	20.7%
8	5	8	299,520	2.03%	2,682,980	18.2%
9	5	8	336,960	1.80%	3,018,340	16.1%
10	5	8	374,400	1.62%	3,353,700	14.5%
11	5	8	411,840	1.47%	3,689,060	13.2%
12	5	8	449,280	1.35%	4,024,420	12.1%
13	5	8	486,720	1.25%	4,359,780	11.2%
14	5	8	524,160	1.16%	4,695,140	10.4%
15	5	8	561,600	1.08%	5,030,500	9.68%
16	5	8	599,040	1.01%	5,365,860	9.08%
17	5	8	636,480	0.95%	5,701,220	8.55%
18	5	8	673,920	0.90%	6,036,580	8.07%
19	5	8	711,360	0.85%	6,371,940	7.65%
20	5	8	748,800	0.81%	6,707,300	7.27%
5	3	6	9360	7.07%	44,704	33.4%
5	4	7	42,240	4.54%	283,601	30.3%
5	5	8	187,200	3.24%	1,676,900	28.9%
5	6	9	829,440	2.50%	9,388,921	28.2%
5	7	10	3,682,560	2.04%	50,337,424	27.8%

References

Aguirregabiria, V. and P. Mira (2007), "Sequential estimation of dynamic discrete games." *Econometrica*, 75, 1–53. [568, 569, 571, 572, 573, 574, 575, 576, 577, 580, 581, 582, 583, 585]

Aguirregabiria, V. and A. Nevo (2013), "Recent developments in empirical IO: Dynamic demand and dynamic games." In *Advances in Economics and Econometrics, Tenth World Congress*, Vol. 3, 53–122, Cambridge University Press, Cambridge, U.K. [579]

Aitchison, J. and S. D. Silvey (1958), "Maximum likelihood estimation of parameters subject to restraints." *Annals of Mathematical Statistics*, 29, 813–828. [574]

Arcidiacono, P. and R. A. Miller (2011), "CCP estimation of dynamic discrete choice models with unobserved heterogeneity." *Econometrica*, 79, 1823–1868. [568, 594]

Bajari, P., C. L. Benkard, and J. Levin (2007), "Estimating dynamic games of imperfect competition." *Econometrica*, 75, 1331–1370. [568, 572, 574, 575]

Ciliberto, F. and E. Tamer (2009), "Market structure and multiple equilibria in airline markets." *Econometrica*, 77, 1791–1828. [572]

Dubé, J.-P., J. T. Fox, and C.-L. Su (2012), "Improving the numerical performance of static and dynamic aggregate discrete choice random coefficients demand estimation." *Econometrica*, 80, 2231–2267. [568]

Gourieroux, C. and A. Monfort (1995), *Statistics and Econometric Models*, Vol. 1. Cambridge University Press, Cambridge, U.K. [574]

Hotz, J. and R. A. Miller (1993), "Conditional choice probabilities and the estimation of dynamic models." *Review of Economic Studies*, 60, 497–529. [574]

Igami, M. (2013), "Estimating the innovator's dilemma: Structural analysis of creative destruction." Working paper, Yale University. [568]

Igami, M. (2014), "Offshoring under oligopoly." Working paper, Yale University. [568]

Kasahara, H. and K. Shimotsu (2012), "Sequential estimation of structural models with a fixed point constraint." *Econometrica*, 80, 2303–2319. [568, 573, 575, 577, 578, 579, 580, 581, 582, 583, 591]

Ortega, J. M. and W. C. Rheinboldt (1970), *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York and London. [578]

Pakes, A., M. Ostrovsky, and S. Berry (2007), "Simple estimators for the parameters of discrete dynamic games, with entry/exit examples." *RAND Journal of Economics*, 38, 373–399. [568, 572, 574, 575, 576, 577]

Pakes, A., J. Porter, K. Ho, and J. Ishii (2011), "Moment inequalities and their applications." Working paper, Harvard University. [572]

Pesendorfer, M. and P. Schmidt-Dengler (2008), "Asymptotic least squares estimators for dynamic games." *Review of Economic Studies*, 75, 901–928. [568, 572, 574, 575]

Pesendorfer, M. and P. Schmidt-Dengler (2010), "Sequential estimation of dynamic discrete games: A comment." *Econometrica*, 78, 833–842. [568, 575, 577]

Rust, J. (1987), "Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher." *Econometrica*, 55, 999–1033. [573]

Sanches, F., D. Silva, and S. Srisuma (2013), "An alternative asymptotic least squares estimator for dynamic games." Working paper, University of São Paulo. [568]

Srisuma, S. (2013), "Minimum distance estimators for dynamic games." *Quantitative Economics*, 4, 549–583. [568]

Su, C.-L. (2014), "Estimating discrete-choice games of incomplete information: Simple static examples." *Quantitative Marketing and Economics*, 12, 167–207. [568, 569, 573, 575, 577]

Su, C.-L. and K. L. Judd (2012), "Constrained optimization approaches to estimation of structural models." *Econometrica*, 80, 2213–2230. [568, 569, 574]

Tamer, E. (2003), "Incomplete simultaneous discrete response with multiple equilibria." *Review of Economic Studies*, 70, 147–165. [572]

Submitted January, 2014. Final version accepted September, 2014.