Constrained Optimization Approaches to Estimation of Structural Models

Che-Lin Su
The University of Chicago
Booth School of Business
Che-Lin.Su@ChicagoBooth.edu

The 2014 Zurich Initiative on Computational Economics
January 28 – February 5, 2014

Outline

1. Estimation of Dynamic Programming Models of Individual Behavior

Outline

1. Estimation of Dynamic Programming Models of Individual Behavior
2. Estimation of Demand Systems

Outline

1. Estimation of Dynamic Programming Models of Individual Behavior
2. Estimation of Demand Systems
3. Estimation of Static Games of Incomplete Information

Outline

1. Estimation of Dynamic Programming Models of Individual Behavior
2. Estimation of Demand Systems
3. Estimation of Static Games of Incomplete Information
4. Estimation of Dynamic Games of Incomplete Information

Part I

Optimization Overview

Unconstrained Optimization: Background

$$\min \{f(x) : x \in \mathbb{R}^n\}$$

- $f : R^n \to R$ smooth (typically $\mathcal{C}^2$)
- $x \in R^n$ finite dimensional (may be large)

Optimality conditions: x^* local minimizer:

$$\nabla f(x^*) = 0$$

Numerical methods: generate a sequence of iterates x_k such that the gradient test

$$\|\nabla f(x_k)\| \le \tau$$

is eventually satisfied; usually $\tau = 1.e - 6$

Warning: Any point x that does NOT satisfy $\|\nabla f(x)\| \le \tau$ should NOT be considered as a "solution" or a candidate for the solution

Did the solver Find a Solution?

Iteration	Func-count	f(x)	Step-size	First-order optimality
0	1	51770.3		5.53e+004
1	2	5165.79	1.80917e-005	1.26e+004
2	3	3604.44	1	9.05e+003
3	4	2482.01	1	6.01e+003
20	22	209.458	1	150
21	23	207.888	1	151
22	24	199.115	1	166
23	25	188.692	1	217
24	26	162.908	1	325
25	27	143.074	1	614
26	28	129.016	1	320
27	29	113.675	1	205
28	30	94.7791	1	184
29	32	75.7777	0.431713	166
30	33	71.4657	1	110
31	34	71.0592	1	55

```
Optimization terminated: relative infinity-norm of gradient
less than options.TolFun.
```

Generic Nonlinear Optimization Problem

Nonlinear Programming (NLP) problem

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) & \text{objective} \\ \text{subject to} & c(x) = 0 & \text{constraints} \\ & x \geq 0 & \text{variables} \end{cases}$$

- $f : R^n \to R$, $c : R^n \to R^m$ smooth (typically $\mathcal{C}^2$)
- $x \in R^n$ finite dimensional (may be large)
- more general $l \leq c(x) \leq u$ possible

Optimality Conditions for NLP

Constraint qualification (CQ)
Linearizations of $c(x) = 0$ characterize all feasible perturbations

x^* local minimizer & CQ holds $\Rightarrow \exists$ multipliers y^*, z^*:

$$\begin{aligned} \nabla f(x^*) - \nabla c(x^*)^T y^* - z^* &= 0 \\ c(x^*) &= 0 \\ X^* z^* &= 0 \\ x^* \geq 0,\ z^* &\geq 0 \end{aligned}$$

where $X^* = \text{diag}(x^*)$, thus $X^* z^* = 0 \iff x_i^* z_i^* = 0$

Solving the FOC for NLP

- Nonlinear equations: $F(w) = 0$, where $w = (x, y, z)$ with $x, z \geq 0$.
- NLP solvers: generate a sequence of iterates w_k such that the test

$$\|\nabla F(w_k)\| \leq \tau \text{ with } x_k \geq 0,\ z_k \geq 0$$

  is eventually satisfied; usually $\tau = 1.e - 6$. Same warning applies.
- Supply exact derivatives: $\nabla f(x), \nabla c(x), \nabla^2 \mathcal{L}(x, y, z)$,
  where is the Lagrangian: $\mathcal{L}(x, y, z) := f(x) - y^T c(x) - z^T x$
- Concerns: NLP is difficult to solve when # of variables and # of constraints are large
- In many applied models, constraint Jacobian $\nabla c(x)$ and Hessian of the Lagragian $\nabla^2 \mathcal{L}(x)$ are sparse
- Modern solvers exploit the sparsity structure of $\nabla c(x)$ and $\nabla^2 \mathcal{L}(x)$

Structural Estimation

- Great interest in estimating models based on economic structure
  - DP models of individual behavior: Rust (1987)
  - Demand Estimation: BLP(1995), Nevo(2000)
  - Nash equilibria of static and dynamic games: AM (2007), BBL (2007), Pakes, Ostrovsky and Berry (2007), Pesendorfer and Schmidt-Dengler (2008)
  - Auctions: Paarsch and Hong (2006), Hubbard and Paarsch (2008)
  - Dynamic stochastic general equilibrium
  - Popularity of structural models in empirical IO and marketing
- Model sophistication introduces computational difficulties
- General belief: Estimation is a major computational challenge because it involves solving the model many times
- Our approach: Formulate structural estimation models as constrained optimization problems and use modern constrained optimization methods and software to solve the models for you

Structural Estimation in Microeconomics

- Single-Agent Dynamic Discrete Choice Models
  - Rust (1987): Bus-Engine Replacement Problem
  - Nested-Fixed Point Problem (NFXP)
  - Su and Judd (2012): Constrained Optimization Approach
- Random-Coefficients Logit Demand Models
  - BLP (1995): Random-Coefficients Demand Estimation
  - Nested-Fixed Point Problem (NFXP)
  - Dubé, Fox and Su (2012): Constrained Optimization Approach
- Estimating Discrete-Choice Games of Incomplete Information
  - Aguirregabiria and Mira (2007): NPL (Recursive 2-Step)
  - Bajari, Benkard and Levin (2007): 2-Step
  - Pakes, Ostrovsky and Berry (2007): 2-Step
  - Pesendorfer and Schmidt-Dengler (2008): 2-Step
  - Pesendorfer and Schmidt-Dengler (2010): comments on AM (2007)
  - Kasahara and Shimotsu (2012): Modified NPL
  - Su (2013), Egesdal, Lai and Su (20013): Constrained Optimization

Optimization and Computation in Structural Estimation

- Optimization and computation often perceived as 2nd-order importance to research agenda
- Typical computational methohd is Nested Fixed-Point procedure: fixed-point calculation embedded in calculation of objective function
  - compute an "equilibrium"
  - invert a model (e.g. non-linearity in disturbance)
  - compute a value function (i.e. dynamic model)
- Mis-use of optimization can lead to the "wrong answer"
  - naively use canned optimization algorithms – e.g., Matlab's fminsearch
  - adjust default-settings of solvers to improve speed not accuracy
  - assume there is a unique fixed-point
  - do NOT check or understand solver's output message!!
    - KNITRO: Locally Optimal Solution Found.
    - FilterSQP: Optimal Solution Found.
    - SNOPT: Optimal Solution Found.
    - Matlab Optimization Toolbox: Optimization terminated ... does NOT tell you much about what happened at the end

First Step in Solving an Estimation Model

- Make sure you have a smooth formulation for the model
  - smooth objective function
  - smooth constraints
- Use the best available NLP solvers!
  - Many free NLP solvers are crappy; they often fail or even worse, can give you wrong solutions
  - Do not attempt to develop numerical algorithms/solvers by yourself
  - You should use solvers developed by "professionals", i.e., numerical optimization people
  - Best NLP solvers: SNOPT (Stanford), KNITRO (Northwestern), Filter-SQP (Argonne), IPOPT (IBM), PATH (UW-Madison)
- Keys to efficient implementation
  - Supply exact 1st and 2nd order derivatives
  - Supply sparsity pattern for constraint Jacobian and Hessian of the Lagragian

Part II

Estimation of Dynamic Programming Models

Rust (1987): Zurcher's Data

Bus #: 5297

events	year	month	odometer at replacement
1st engine replacement	1979	June	242400
2nd engine replacement	1984	August	384900

year	month	odometer reading
1974	Dec	112031
1975	Jan	115223
1975	Feb	118322
1975	Mar	120630
1975	Apr	123918
1975	May	127329
1975	Jun	130100
1975	Jul	133184
1975	Aug	136480
1975	Sep	139429

Zurcher's Bus Engine Replacement Problem

- Each bus comes in for repair once a month
- Bus manager sees
  - x_t: mileage at time t since last engine overhaul
  - $\varepsilon_t = [\varepsilon_t(d_t = 0), \varepsilon_t(d_t = 1)]$: other state variable
- Bus manager chooses between overhaul and ordinary maintenance

$$d_t = \begin{cases} 1, & \text{replacing the engine;} \\ 0, & \text{performing regular maintenance.} \end{cases}$$

- Utility per period $u(x_t, d_t, \varepsilon_t; \theta^c, RC) = \nu(x_t, d_t; \theta^c, RC) + \varepsilon_t(d_t)$ where

$$\nu(x_t, d_t, \theta^c, RC) = \begin{cases} -c(x_t, \theta^c) & \text{if} \quad d_t = 0 \\ -(RC + c(0, \theta^c) & \text{if} \quad d_t = 1 \end{cases}$$

  - $c\,(x; \theta^c)$: expected operating costs per period at mileage x
  - RC: the expected replacement cost to install a new engine, net of any scrap value of the old engine
  - The mileage x is reset to 0 after the engine replacement

Zurcher's Bus Engine Replacement Problem

- Given (x_t, ε_t), the bus manager solves the DP:

$$\max_{\{d_t, d_{t+1}, d_{t+2}, \ldots\}} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} u(x_\tau, d_\tau, \varepsilon_\tau; \theta^c, RC)\right]$$

  - The expectation $\mathbb{E}$ is taken over the state transition probability $p(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, d_t; \theta^p)$
- Value function

$$V(x_t, \varepsilon_t) = \max_{\{d_t, d_{t+1}, d_{t+2}, \ldots\}} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} u(x_\tau, d_\tau, \varepsilon_\tau; \theta^c, RC)\right]$$

- Econometrician
  - Observes mileage x_t and decision d_t, but not cost
  - Assumes extreme value distribution for $\varepsilon_t(d_t)$
- Structural parameters to be estimated: $\theta = (\theta^c, RC, \theta^p)$
  - Coefficients of operating cost function; e.g., $c(x, \theta^c) = \theta_1^c x + \theta_2^c x^2$
  - Overhaul cost RC
  - state transition probability $p(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, d_t; \theta^p)$

Zurcher's Bus Engine Replacement Problem

- Bellman equation

$$V(x,\varepsilon) = \max_{d} \left\{ \nu(x,d;\theta^c,RC) + \varepsilon(d) + \beta \int_{x'} \int_{\epsilon'} V(x',\varepsilon') p(x',\varepsilon'|x,\varepsilon,d;\theta^p) dx' d\varepsilon' \right\}$$

- Conditional Independence (CI) Assumption:

$$p(x',\varepsilon'|x,\varepsilon,d;\theta^p) = p_2(\varepsilon'|x';\theta_2^p) p_3(x'|x,d;\theta_3^p)$$

- Expected value function

$$EV(x) = \int_{\epsilon} V(x,\varepsilon) p_2(\varepsilon'|x';\theta_2^p)$$

- Choice-specific expected value function

$$EV(x,d) = \nu(x,d;\theta^c,RC) + \varepsilon(d) + \beta \int_{x'} EV(x) p_3(x'|x,d;\theta_3^p) dx'$$

Zurcher's Bus Engine Replacement Problem

- Assume type-1 extreme value distribution for $\varepsilon = [\varepsilon(0), \varepsilon(1)]$
- Conditional choice probability

$$P(d|x;\theta) = \frac{\exp\left[\nu(x,d;\theta^c,RC) + \beta EV(x,d)\right]}{\sum\limits_{d' \in \{0,1\}} \exp\left[\nu(x,d';\theta^c,RC) + \beta EV(x,d')\right]}$$

- Choice-specific expected value function

$$EV(x,d) = \int_{x'=0}^{\infty} \log\left\{\sum_{d'\in\{0,1\}} \exp\left[\nu(x',d';\theta^c,RC) + \beta EV(x',d')\right]\right\} p_3(dx'|x,d,\theta_3^p)$$

Zurcher's Bus Engine Replacement Problem

- Discretize the mileage state space x into K grid points $\hat{\mathbf{x}} = \{\hat{x}_1, \ldots, \hat{x}_K\}$ with $\hat{x}_1 = 0$
- Mileage transition probability: for $j = 1, \ldots, J$

$$p_3(x'|\hat{x}_k, d, \theta_3^p) = \begin{cases} \text{Pr}\{x' = \hat{x}_{k+j}|\theta_3^p\}, & \text{if } d = 0 \\ \text{Pr}\{x' = \hat{x}_{1+j}|\theta_3^p\}, & \text{if } d = 1 \end{cases}$$

  - Mileage in the next period x' can move up at most J grid points
- Choice-specific expected value function for $\hat{x} \in \hat{\mathbf{x}}$

$$EV(\hat{x}, d) = \sum_{j=0}^{J} \log \left\{ \sum_{d' \in \{0,1\}} \exp\left[\nu(x', d'; \theta^c, RC) + \beta EV(x', d')\right] \right\} p_3(x'|\hat{x}, d, \theta_3^p)$$

Zurcher's Bus Engine Replacement Problem

- Data: time series $(x_t, d_t)_{t=1}^T$
- Likelihood function

$$L(\theta) = \prod_{t=2}^{T} P(d_t | x_t, \theta^c, RC) p_3(x_t | x_{t-1}, d_{t-1}, \theta_3^p)$$

with $P(d|x, \theta^c, RC) = \dfrac{\exp\{\nu(x, d; \theta^c, RC) + \beta EV_\theta(x, d)\}}{\sum_{d' \in \{0,1\}} \exp\{\nu(x, d'; \theta^c, RC) + \beta EV_\theta(x', d)\}}$

$$EV_\theta(x, d) = T_\theta(EV_\theta)(x, d)$$

$$\equiv \sum_{j=0}^{J} \log \left\{ \sum_{d' \in \{0,1\}} \exp\left[\nu(x', d'; \theta^c, RC) + \beta EV(x', d')\right] \right\} p_3(x' | \hat{x}, d, \theta_3^p)$$

Nested Fixed Point Algo: Rust (1987)

- Outer loop: Solve likelihood

$$\max_{\theta \geq 0} L(\theta) = \prod_{t=2}^{T} P(d_t | x_t, \theta^c, RC) p_3(x_t | x_{t-1}, d_{t-1}, \theta_3^p)$$

  - Convergence test: $\|\nabla_\theta \mathcal{L}(\theta)\| \leq \epsilon_{out}$

- Inner loop: Compute expected value function EV_θ for a given θ
  - EV_θ is the implicit expected value function defined by the Bellman equation or the fixed point function

$$EV_\theta = T_\theta(EV_\theta)$$

  - Convergence test: $\|EV_\theta^{k+1} - EV_\theta^{k}\| \leq \epsilon_{in}$
  - Rust started with contraction iterations and then switched to Newton iterations

Smooth Objective Function?

- Is the ML objective function $\mathcal{L}(\theta)$ smooth (differentiable w.r.t. θ) ?
  - $L(\theta) = \prod_{t=2}^{T} P(d_t|x_t, \theta^c, RC) p_3(x_t|x_{t-1}, d_{t-1}, \theta_3^p)$
  - $P(d|x, \theta^c, RC) = \dfrac{\exp\{\nu(x, d; \theta^c, RC) + \beta EV_\theta(x, d)\}}{\sum_{d' \in \{0,1\}} \exp\{\nu(x, d'; \theta^c, RC) + \beta EV_\theta(x', d)\}}$
  - Is EV_θ differentiable w.r.t. θ ?
    Yes, because $T_\theta(EV_\theta)$ is a contraction mapping(!)
- Is the "approximated" ML objective function $L(\theta, \epsilon_{in})$ smooth (differentiable w.r.t. θ) ?
  - Is $EV_\theta(\epsilon_{\text{in}})$ differentiable w.r.t. θ or w.r.t. ϵ_{in} ?

Concerns with NFXP – Dubé Fox and Su (2011)

- Inner-loop error propagates into outer-loop function and derivatives
- NFXP needs to solve inner-loop exactly for each vector of parameters
  - to accurately compute the search direction for the outer loop
  - to accurately evaluate derivatives for the outer loop
  - for the outer loop to converge
- Stopping rules: choosing inner-loop and outer-loop tolerances
  - inner-loop can be slow: contraction mapping is linearly convergent
  - tempting to loosen inner loop tolerance ϵ_{in} used
    - often see $\epsilon_{in} = 1.e - 6$ or higher
  - outer loop may not converge with loose inner loop tolerance
    - check solver output message
    - tempting to loosen outer loop tolerance ϵ_{in} to promote convergence
    - often see $\epsilon_{out} = 1.e - 3$ or higher
- Rust's implementation of NFXP was careful and correct
  - $\epsilon_{in} = 1.e - 13$
  - finished the inner-loop with Newton's method

Stopping Rules – Dubé Fox and Su (2011)

- Notations:
  - $L(\theta, \epsilon_{in})$: the programmed outer loop objective function with ϵ_{in}
- Analytic derivatives $\nabla_\theta L(\theta, \epsilon_{in})$ are provided: $\epsilon_{out} = O(\frac{\beta}{1-\beta}\epsilon_{in})$
- Finite-difference derivatives are used: $\epsilon_{out} = O(\sqrt{\frac{\beta}{1-\beta}\epsilon_{in}})$

Constrained Optimization for Solving Zucher Model

- Form augmented likelihood function for data $X = (x_t, d_t)_{t=1}^T$

$$\mathcal{L}\left(\theta, EV; X\right) = \prod_{t=2}^{T} P(d_t|x_t, \theta^c, RC) p(x_t|x_{t-1}, d_{t-1}, \theta^p)$$

with $P(d|x, \theta^c, RC) = \dfrac{\exp\{\nu(x, d; \theta^c, RC) + \beta EV(x, d)\}}{\sum_{d' \in \{0,1\}} \exp\{\nu(x, d'; \theta^c, RC) + \beta EV(x, d')\}}$

- Rationality and Bellman equation imposes a relationship between θ and EV

$$EV = T\left(EV, \theta\right)$$

- Solve the constrained optimization problem

$$\begin{array}{ll} \max\limits_{(\theta, EV)} & \mathcal{L}\left(\theta, EV; X\right) \\ \text{subject to} & EV = T\left(EV, \theta\right) \end{array}$$

Equivalent Reformulation?

- ML-NFXP:

$$\max_{\theta \geq 0} L(\theta) = \prod_{t=2}^{T} P(d_t | x_t, \theta^c, RC) p_3(x_t | x_{t-1}, d_{t-1}, \theta_3^p)$$

- ML-Constrained Optimization:

$$\begin{array}{ll} \max\limits_{(\theta, EV)} & \mathcal{L}\left(\theta, EV; X\right) \\ \text{subject to} & EV = T\left(EV, \theta\right) \end{array}$$

- Are these two formulations equivalent? Proof?
- Are the first-order conditions of these two formulations equivalent? Proof?

Monte Carlo: Rust's Table X - Group 1,2, 3

- Fixed point dimension: 175
- Maintenance cost function: $c(x, \theta^c) = 0.001 * \theta_1^c * x$
- Mileage transition: stay or move up at most 4 grid points
- True parameter values:
    - $\theta_1^c = 2.457$
    - $RC = 11.726$
    - $(\theta_{30}^p, \theta_{31}^p, \theta_{32}^p, \theta_{33}^p) = (0.0937, 0.4475, 0.4459, 0.0127)$
    - Solve for EV at the true parameter values
- Simulate 250 datasets of monthly data for 10 years and 50 buses
- Estimation implementations: `DP\MCscript\RustBusMLETableX_MC.m`
    - MPEC1: AMPL/Knitro (with 1st- and 2nd-order derivative)
    - MPEC2: Matlab/ktrlink (with 1st-order derivatives)
    - NFXP: Matlab/ktrlink (with 1st-order derivatives)
    - 5 re-start in each of 250 replications

Monte Carlo: Rust's Table X - AMPL Code

- AMPL files:
  - AMPL Model File: `RustBusMLETableX.mod`
  - AMPL Data File: `RustBusMLETableX.dat`
  - AMPL Command File: `RustBusMLETableX.run`
  - Remember to change the path to the KNITRO (or other) solver on your computer
- Solve an optimization problem in AMPL:
  - ampl:
    ampl:
    ampl: include RustBusMLETableX.run

AMPL Model: RustBusMLETableX.mod

```
# SET UP THE MODEL and DATA #
param nBus;              # number of buses in the data
set B := 1..nBus;        # B is the index set of buses
param nT;  # number of periods in the data
set T := 1..nT;          # T is the vector of time indices

# Define the state space used in the dynamic programming part
param    N;            # number of discrete grids in the mileage state
set X := 1..N;           # X is the index set of states
param x {i in X} := i;   #  x[i] denotes state i;

# In this example, M = 5: the bus mileage reading in the next period can
# either stay in current state or can move up to 4 states
param M;

# Define discount factor. We fix beta since it can't be identified.
param beta;        # discount factor

# Data: (xt, dt)
param dt {t in T, b in B};  # decision of bus b at time t
param xt {t in T, b in B};    # mileage (state) of bus b at time t

# END OF MODEL and DATA SETUP #
```

AMPL Model: RustBusMLETableX.mod

```
# DEFINING STRUCTURAL PARAMETERS and ENDOGENOUS VARIABLES TO BE SOLVED #
# Parameters for (linear) cost function
#    c(x, thetaCost) = 0.001*thetaCost*x ;
var thetaCost >= 0;

# thetaProbs[i] defines transition probability that mileage in next period moves up
# M=5 in this example.
var thetaProbs {1..M} >= 0;

# Replacement cost
var RC >= 0;

# Define variables for specifying initial parameter values
var iniRC;
var inithetaCost;
var iniEV;

# DECLARE EQUILIBRIUM CONSTRAINT VARIABLES
# The NLP approach requires us to solve equilibrium constraint variables
var EV {X};         # Expected Value Function of each state

# END OF DEFINING STRUCTURAL PARAMETERS AND ENDOGENOUS VARIABLES #
```

AMPL Model: RustBusMLETableX.mod

```
#  Define auxiliary variables to economize on expressions

#  Create Cost variable to represent the cost function;
#  Cost[i] is the cost of regular maintenance at x[i].
var Cost {i in X} = 0.001*thetaCost*x[i];

#  Let CbEV[i] represent - Cost[i] + beta*EV[i];
#  this is the expected payoff at x[i] if regular maintenance is chosen
var CbEV {i in X} = - Cost[i] + beta*EV[i];

#  Let PayoffDiff[i] represent -CbEV[i] - RC + CbEV[1];
#  this is the difference in expected payoff at x[i] between engine replacement
#  and regular maintenance
var PayoffDiff {i in X} = -CbEV[i] - RC + CbEV[1];

#  Let ProbRegMaint[i] represent 1/(1+exp(PayoffDiff[i]));
#  this is the probability of performing regular maintenance at state x[i];
var ProbRegMaint {i in X} = 1/(1+exp(PayoffDiff[i]));

# BellmanViola represents violation of the Bellman equations.
var BellmanViola {i in 1..(N-M+1)} = sum {j in 0..(M-1)} log(exp(CbEV[i+j])
+ exp(-RC + CbEV[1]))* thetaProbs[j+1] - EV[i];
```

AMPL Model: RustBusMLETableX.mod

```
# DEFINE OBJECTIVE FUNCTION AND CONSTRAINTS #

# Define the objective: Likelihood function
#   The likelihood function contains two parts:
#   First is the likelihood that the engine is replaced at the state at time t;
#   Second is the likelihood that the observed transition between
#   t-1 and t would have occurred.

maximize Likelihood:
sum {t in 2..nT, b in B} log(dt[t,b]*(1-ProbRegMaint[xt[t,b]])
            + (1-dt[t,b])*ProbRegMaint[xt[t,b]])
+ sum {t in 2..nT, b in B} log(dt[t-1,b]*(thetaProbs[xt[t,b]-1+1])
            + (1-dt[t-1,b])*(thetaProbs[xt[t,b]-xt[t-1,b]+1]));
```

AMPL Model: RustBusMLETableX.mod

```
subject to          #  Define the constraints
#  Bellman equation for states below N-M
Bellman_1toNminusM {i in X: i <= N-(M-1)}:
      EV[i] = sum {j in 0..(M-1)}
                log(exp(CbEV[i+j])+ exp(-RC + CbEV[1]))* thetaProbs[j+1];

#  Bellman equation for states above N-M
# (we adjust transition probabilities to keep state in [xmin, xmax])
Bellman_LastM {i in X: i > N-(M-1) and i <= N-1}:
EV[i] = (sum {j in 0..(N-i-1)}
log(exp(CbEV[i+j])+ exp(-RC + CbEV[1]))*thetaProbs[j+1])
+ (1- sum {k in 0..(N-i-1)} thetaProbs[k+1])*log(exp(CbEV[N])+ exp(-RC + CbEV[1]));

#  Bellman equation for state N
Bellman_N:  EV[N] = log(exp(CbEV[N])+ exp(-RC + CbEV[1]));

#  The probability parameters in transition process must add to one
   Probability: sum {i in 1..M} thetaProbs[i] = 1;

#  Put bound on EV; this should not bind.
# This is a cautionary step to preventing algorithm from diverging
EVBound {i in X}: EV[i] <= 500;
```

AMPL Model: RustBusMLETableX.mod

```
# DEFINE THE MLE OPTIMIZATION PROBLEM #

# Name the problem
problem MPECZurcher:

# Choose the objective function
Likelihood,

# List the variables
EV, RC, thetaCost, thetaProbs, Cost, CbEV, PayoffDiff,
ProbRegMaint, BellmanViola,

# List the constraints
Bellman_1toNminusM,
Bellman_LastM,
Bellman_N,
Probability,
EVBound;

# END OF DEFINING THE MLE OPTIMIZATION PROBLEM
```

AMPL/KNITRO Output

```
KNITRO 7.7.0: alg=1
opttol=1.0e-6
feastol=1.0e-6

Problem Characteristics
-----------------------
Objective goal:  Maximize
Number of variables:                  182
    bounded below:                      7
    bounded above:                    175
    bounded below and above:            0
    fixed:                              0
    free:                               0
Number of constraints:                176
    linear equalities:                  1
    nonlinear equalities:             175
    linear inequalities:                0
    nonlinear inequalities:             0
    range:                              0
Number of nonzeros in Jacobian:      2255
Number of nonzeros in Hessian:       1585
```

AMPL/KNITRO Output

Iter	Objective	FeasError	OptError	||Step||	CGits
0	-9.865625e+03	8.742e-01			
1	-1.357273e+04	8.704e-01	4.977e+00	2.645e-01	6
2	-1.302591e+04	8.197e-01	2.483e+01	4.663e-01	1
3	-1.415726e+04	8.194e-01	4.374e+01	4.807e-02	14
4	-1.431110e+04	8.193e-01	4.088e+01	5.238e-03	7
5	-1.434829e+04	8.193e-01	3.051e+01	1.149e-03	6
6	-1.435348e+04	8.193e-01	1.168e+02	1.258e-04	7
7	-1.198544e+04	3.798e+00	1.078e+02	1.755e+01	0
8	-6.787923e+03	4.031e+00	3.931e+01	3.594e+01	0
9	-6.263886e+03	6.676e-01	5.037e+01	1.472e+01	0
10	-6.149228e+03	2.292e-01	7.276e+00	1.260e+01	0
11	-6.117234e+03	7.604e-02	1.125e+00	1.238e+01	0
12	-6.105937e+03	3.902e-03	1.060e+00	3.874e+01	0
13	-6.099758e+03	2.340e-03	5.538e-01	4.062e+01	0
14	-6.097581e+03	1.574e-03	3.145e-01	3.791e+01	0
15	-6.097192e+03	4.259e-04	4.583e-02	2.260e+01	0
16	-6.097170e+03	1.641e-05	1.226e-03	5.038e+00	0
17	-6.097170e+03	2.087e-08	1.554e-06	1.916e-01	0

EXIT: Locally optimal solution found.

AMPL/KNITRO Output

```
Final Statistics
-----------------
Final objective value                   =  -6.09716956266980e+03
Final feasibility error (abs / rel) =   2.09e-08 / 2.09e-08
Final optimality error  (abs / rel) =   1.55e-06 / 3.52e-08
# of iterations                         =         17
# of CG iterations                      =         41
# of function evaluations               =         45
# of gradient evaluations               =         18
# of Hessian evaluations                =         17
Total program time (secs)               =       0.24043 (      0.227 CPU time)
Time spent in evaluations (secs)        =       0.20118

===============================================================================

KNITRO 7.0.0: Locally optimal solution.
objective -6097.169563; feasibility error 2.09e-08
17 iterations; 45 function evaluations
_solve_time = 0.269684
```

Monte Carlo: $\beta = 0.975$ and 0.980

β	Imple.	Parameters						MSE
		RC	θ_1^c	θ_{30}^p	θ_{31}^p	θ_{32}^p	θ_{33}^p	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.975	MPEC1	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.213	2.606	0.0943	0.4473	0.4445	0.0127	3.123
		(1.617)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
0.980	MPEC1	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.139	2.579	0.0943	0.4473	0.4455	0.0127	2.866
		(1.571)	(0.459)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–

Monte Carlo: $\beta = 0.985$ and 0.990

β	Imple.	Parameters						MSE
		RC	θ_1^c	θ_{31}^p	θ_{32}^p	θ_{33}^p	θ_{34}^p	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.985	MPEC1	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.021	2.544	0.0943	0.4473	0.4455	0.0127	2.136
		(1.368)	(0.411)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
0.990	MPEC1	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–

Monte Carlo: $\beta = 0.995$

β	Imple.	Parameters						MSE
		RC	θ_1^c	θ_{31}^p	θ_{32}^p	θ_{33}^p	θ_{34}^p	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.995	MPEC1	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–

Monte Carlo: Numerical Performance

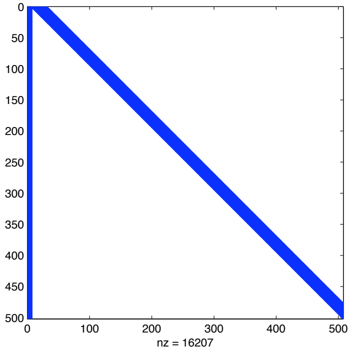
β	Imple.	Runs Conv.	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Contrac. Mapping Iter.
0.975	MPEC1	1240	0.13	12.8	17.6	–
	MPEC2	1247	7.9	53.0	62.0	–
	NFXP	998	24.6	55.9	189.4	134,748
0.980	MPEC1	1236	0.15	14.5	21.8	–
	MPEC2	1241	8.1	57.4	70.6	–
	NFXP	1000	27.9	55.0	183.8	162,505
0.985	MPEC1	1235	0.13	13.2	19.7	–
	MPEC2	1250	7.5	55.0	62.3	–
	NFXP	952	42.2	61.7	227.3	265,827
0.990	MPEC1	1161	0.19	18.3	42.2	–
	MPEC2	1248	7.5	56.5	65.8	–
	NFXP	935	70.1	66.9	253.8	452,347
0.995	MPEC1	965	0.14	13.4	21.3	–
	MPEC2	1246	7.9	59.6	70.7	–
	NFXP	950	111.6	58.8	214.7	748,487

Observations

- MPEC
  - In MPEC/AMPL, problems are solved very quickly.
  - The likelihood function, the constraints, and their first-order and second-order derivatives are evaluated only around 20 times
  - Constraints (Bellman Eqs) are NOT solved exactly in most iterations
    - No need to resolve the fixed-point equations for every guess of structural parameters
  - Quadratic convergence is observed in the last few iterations; in contrast, NFXP is linearly convergent (or super-linear at best)
- In NFXP, the Bellman equations are solved around 200 times and evaluated between 134,000 and 750,000 times

Advantages of Constrained Optimization

- Newton-based methods are locally quadratic convergent
- Two key factors in efficient implementations:
  - Provide analytic-derivatives – huge improvement in speed
  - Exploit sparsity pattern in constraint Jacobian – huge saving in memory requirement



Part III

Random-Coefficients Demand Estimation

Random-Coefficients Logit Demand: BLP (1995)

- Berry, Levinsohn and Pakes (BLP, 1995) consists of an economic model and a GMM estimator
- Demand estimation with a large number of differentiated products
  - characteristics approach
  - applicable when only aggregate market share data available
  - flexible substitution patterns / price elasticities
  - control for price endogeneity
- Computational algorithm to construct moment conditions from a non-linear model
- Useful for measuring market power, welfare, optimal pricing, etc.
- Used extensively in empirical IO and marketing: Nevo (2001), Petrin (2002), Dubé (2003–2009), etc.

Random-Coefficients Logit Demand

- Utility of consumer i from purchasing product j in market t

$$u_{ijt} = \beta_i^0 + x_{jt}\beta_i^x - \beta_i^p p_{jt} + \xi_{jt} + \varepsilon_{ijt}$$

- product characteristics: x_{jt}, p_{jt}, ξ_{jt}
  - x_{jt}, p_{jt} observed; $cov(\xi_{jt}, p_{jt}) \neq 0$
  - ξ_{jt}: not observed – not in data
- β_i: random coefficients/individual-specific taste to be estimated
  - Distribution: $\beta_i \sim F_\beta(\beta; \theta)$
  - BLP's statistical goal: estimate θ in parametric distribution
- error term ε_{ijt}: Type I E.V. shock (i.e., Logit)
- Consumer i picks product j if $u_{ijt} \geq u_{ij't}, \quad \forall j' \neq j$

Market Share Equations

- Predicted market shares

$$s_j(x_t, p_t, \xi_t, ; \theta) = \int_{\left\{\beta_i, \varepsilon_j | u_{ijt} \geq u_{ij't}, \forall j' \neq j\right\}} dF_\beta(\beta; \theta) dF_\varepsilon(\varepsilon)$$

- With logit errors ε

$$s_j(x_t, p_t, \xi_t, ; \theta) = \int_\beta \frac{\exp(\beta^0 + x_{jt}\beta^x - \beta^p p_{jt} + \xi_{jt})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x_{kt}\beta^x - \beta^p p_{kt} + \xi_{kt})} dF_\beta(\beta; \theta)$$

- Simulate numerical integral

$$\hat{s}_j(x_t, p_t, \xi_t, ; \theta) = \frac{1}{ns} \sum_{r=1}^{ns} \frac{\exp(\beta^{0r} + x_{jt}\beta^{xr} - \beta^{pr} p_{jt} + \xi_{jt})}{1 + \sum_{k=1}^{J} \exp(\beta^{0r} + x_{kt}\beta^{xr} - \beta^{pr} p_{kt} + \xi_{kt})}$$

- Market share equations

$$\hat{s}_j(x_t, p_t, \xi_t, ; \theta) = S_{jt}, \; \forall j \in J, t \in T$$

Random-Coefficients Logit Demand: GMM Estimator

- Assume $E\left[\xi_{jt} z_{jt} | z_{jt}\right] = 0$ for some vector of instruments z_{jt}
  - Empirical analog $g\left(\theta\right) = \frac{1}{TJ} \sum_{t,j} \xi_{jt}(\theta)' z_{jt}$
- Data: $\left\{\left(x_{jt}, p_{jt}, S_{jt}, z_{jt}\right)_{j \in J, t \in T}\right\}$
- Minimize GMM objective function

$$Q(\theta) = g\left(\theta\right)' W g\left(\theta\right)$$

- Cannot compute $\xi_{jt}(\theta)$ analytically
  - "Invert" ξ_t from system of predicted market shares numerically

$$\begin{array}{rrcl} & S_t & = & s\left(x_t, p_t, \xi_t; \theta\right) \\ \Rightarrow & \xi_t\left(\theta\right) & = & s^{-1}\left(x_t, p_t, S_t; \theta\right) \end{array}$$

- BLP show the inversion of share equations for $\xi(\theta)$ is a contraction-mapping

BLP/NFXP Estimation Algorithm

- Outer loop: $\min_{\theta} g(\theta)' W g(\theta)$
  - Guess θ parameters to compute $g(\theta) = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \xi_{jt}(\theta)' zjt$
  - Stop when $\|\nabla_\theta (g(\theta)' W g(\theta))\| \leq \epsilon_{\text{out}}$

BLP/NFXP Estimation Algorithm

- Outer loop: $\min_{\theta} g(\theta)' W g(\theta)$
  - Guess θ parameters to compute $g(\theta) = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \xi_{jt}(\theta)' zjt$
  - Stop when $\|\nabla_\theta (g(\theta)' W g(\theta))\| \leq \epsilon_{\text{out}}$
- Inner loop: compute $\xi_t(\theta)$ for a given θ
  - Solve $s(x_t, p_t, \xi_t; \theta) = S_{\cdot t}$ for ξ by contraction mapping:

$$\xi_t^{h+1} = \xi_t^h + \log S_t - \log s(x_t, p_t, \xi_t; \theta)$$

  - Stop when $\|\xi_{\cdot t}^{h+1} - \xi_{\cdot t}^h\| \leq \epsilon_{\text{in}}$
  - Denote the approximated demand shock by $\xi(\theta, \epsilon_{\text{in}})$
- Stopping rules: need to choose tolerance/stopping criterion for both inner loop (ϵ_{in}) and outer loop (ϵ_{out})

Smooth Objective Function?

Is the GMM objective function $Q(\xi(\theta))$ smooth (differentiable w.r.t. θ) ?

- $Q(\xi(\theta)) = \xi(\theta)' Z W Z' \xi(\theta)$
- Is $\xi(\theta)$ differentiable w.r.t. θ ?
-

Is the "approximated" GMM objective function $Q(\xi(\theta; \epsilon_{in}))$ smooth (differentiable w.r.t. θ) ?

- $Q(\xi(\theta, \epsilon_{\text{in}})) = \xi(\theta, \epsilon_{\text{in}})' Z W Z' \xi(\theta, \epsilon_{\text{in}})$
- Is $\xi(\theta, \epsilon_{\text{in}})$ differentiable w.r.t. θ or w.r.t. ϵ_{in} ?

Our Concerns with NFP/BLP

- Inefficient amount of computation
  - we only need to know $\xi(\theta)$ at the true θ
  - NFP solves inner-loop exactly each stage of parameter search
  - evaluating $s(x_t, p_t, \xi_t; \theta)$ thousands of times in the contraction mapping
- Stopping rules: choosing inner-loop and outer-loop tolerances
  - inner-loop can be slow (especially for bad guesses of θ): linear convergence at best
  - tempting to loosen inner loop tolerance ϵ_{in} used
    - often see $\epsilon_{in} = 1.e-6$ or higher
  - outer loop may not converge with loose inner loop tolerance
    - check solver output message; see Knittel and Metaxoglou (2008)
    - tempting to loosen outer loop tolerance ϵ_{in} to promote convergence
    - often see $\epsilon_{out} = 1.e-3$ or higher
- Inner-loop error propagates into outer-loop

Knittel and Metaxoglou (2013)

- Perform extensive numerical studies on BLP/NFXP algorithms with two data sets
  - 10 free solvers and 50 starting points for each solver
- Find that convergence may occur at a number of local extrema, at saddles and in regions of the ob jective function where the First-Order Conditions are not satisfied.
- Furthermore, parameter estimates and measures of market performance, such as price elasticities, exhibit notable variation (two orders of magnitude) depending on the combination of the algorithm and starting values in the optimization exercise at hand
- Recall the optimization output that you saw earlier

Analyzing BLP/NFXP Algorithm

- Let L be the Lipschitz constant of the inner-loop contraction mapping
- Numerical Errors in GMM function and gradient

$$\begin{aligned} \left|Q\left(\xi(\theta,\epsilon_{\text{in}})\right)-Q\left(\xi(\theta,0)\right)\right| &= O\left(\frac{L}{1-L}\epsilon_{\text{in}}\right) \\ \left\|\nabla_\theta Q\left(\xi\left(\theta\right)\right)\big|_{\xi=\xi(\theta,\epsilon_{in})}-\nabla_\theta Q\left(\xi\left(\theta\right)\right)\big|_{\xi=\xi(\theta,0)}\right\| &= O\left(\frac{L}{1-L}\epsilon_{\text{in}}\right) \end{aligned}$$

- Ensuring convergence: $\epsilon_{\text{out}} = O(\frac{L}{1-L})\epsilon_{\text{in}}$

Errors in Parameter Estimates

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \left\{Q\left(\xi(\theta, 0)\right)\right\} \\ \hat{\theta} &= \arg\max_{\theta} \left\{Q\left(\xi(\theta, \epsilon_{\text{in}})\right)\right\}\end{aligned}$$

- Finite sample error in parameter estimates

$$O\left(\left\|\hat{\theta}-\theta^*\right\|^2\right) \leq \left|Q\left(\xi(\hat{\theta}, \epsilon_{\text{in}})\right) - Q\left(\xi(\theta^*, 0)\right)\right| + O\left(\frac{L}{1-L}\epsilon_{\text{in}}\right)$$

- Large sample error in parameter estimates

$$\begin{aligned}\left\|\hat{\theta}-\theta^0\right\| &\leq \left\|\hat{\theta}-\theta^*\right\| + \left\|\theta^*-\theta^0\right\| \\ &\leq \sqrt{\left|Q\left(\xi(\hat{\theta}, \epsilon_{\text{in}})\right) - Q\left(\xi(\theta^*, 0)\right)\right| + O\left(\tfrac{L}{1-L}\epsilon_{\text{in}}\right)} + O\left(1/\sqrt{T}\right)\end{aligned}$$

Numerical Experiment: 100 different starting points

- 1 dataset: 75 markets, 25 products, 10 structural parameters
  - NFP tight: $\epsilon_{in} = 1.e{-}10$; $\epsilon_{out} = 1.e{-}6$
  - NFP loose inner: $\epsilon_{in} = 1.e{-}4$; $\epsilon_{out} = 1.e{-}6$
  - NFP loose both: $\epsilon_{in} = 1.e{-}4$; $\epsilon_{out} = 1.e{-}2$

GMM objective values

Starting point	NFXP tight	NFXP loose inner	NFXP loose both
1	$4.3084e-02$	Fail	$7.9967e+01$
2	$4.3084e-02$	Fail	$9.7130e-02$
3	$4.3084e-02$	Fail	$1.1873e-01$
4	$4.3084e-02$	Fail	$1.3308e-01$
5	$4.3084e-02$	Fail	$7.3024e-02$
6	$4.3084e-02$	Fail	$6.0614e+01$
7	$4.3084e-02$	Fail	$1.5909e+02$
8	$4.3084e-02$	Fail	$2.1087e-01$
9	$4.3084e-02$	Fail	$6.4803e+00$
10	$4.3084e-02$	Fail	$1.2271e+03$

Main findings: Loosening tolerance leads to non-convergence

- Check optimization exit flags!
- Solver does NOT produce a local optimum with loose tolerances!

Constrained Optimization Applied to BLP

- Constrained optimization formulation

$$\begin{array}{ll} \min\limits_{(\theta,\xi)} & \xi^T Z W Z^T \xi \\ \text{subject to} & s(\xi,\theta) = S \end{array}$$

- Advantages:
  - No need to worry about setting up two tolerance levels
  - No inner-loop errors propagated into parameter estimates
  - Easy to code in AMPL and to access good NLP solvers
  - AMPL provides analytic derivatives
  - AMPL analyzes sparsity structure of constraint Jacobian
  - Fewer iterations/function evaluations with first-order and second-order derivatives information
  - Share equations only need to be hold at the solution
- Bad news: Hessian of the Lagrangian is dense

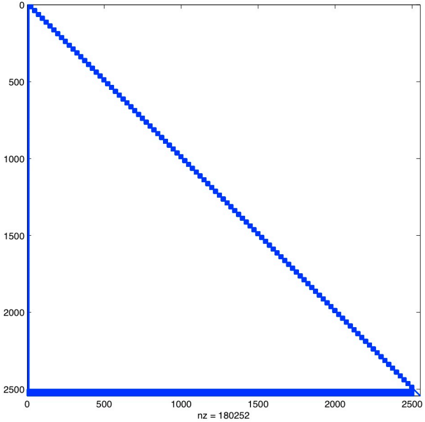
Exploiting Symmetry and Sparsity in the Hessian

- By adding additional variable g and constraint $Z^T\xi = g$

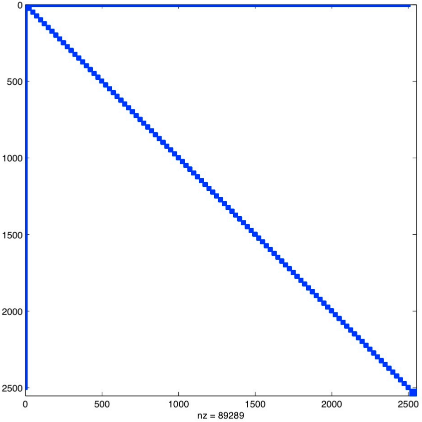
$$\begin{array}{rl} \min\limits_{(\theta,\xi,g,)} & g^T W g \\ \text{subject to} & s(\delta;\theta_2) = S \\ & Z^T\xi = g \end{array}$$

- Advantages:
  - The Hessian of the objective function is now sparse
  - Increasing the sparsity $\Rightarrow$ huge saving on memory

Sparsity Pattern of Constraint Jacobian $\nabla c(x)$



Sparsity Pattern of Hessian $\nabla^2 \mathcal{L}(x, y, z)$



AMPL Model: MPEC_BLP.mod

```
param ns ;    # := 20 ;      # number of simulated "individuals" per market
param nmkt ;  # := 94 ;      # number of markets
param nbrn ;  # := 24 ;      # number of brands per market
param nbrnPLUS1 := nbrn+1;   # number of products plus outside good
param nk1 ;   # := 25;       # of observable characteristics
param nk2 ;   # := 4 ;       # of observable characteristics
param niv ;   # := 21 ;      # of instrument variables
param nz := niv-1 + nk1 -1;  # of instruments including iv and X1
param nd ;    # := 4 ;       # of demographic characteristics

set S := 1..ns ;  # index set of individuals
set M := 1..nmkt ;       # index set of market
set J := 1..nbrn ;       # index set of brand (products), including outside good
set MJ := 1..nmkt*nbrn;  # index of market and brand
set K1 := 1..nk1 ;       # index set of product observable characteristics
set K2 := 1..nk2 ;       # index set of product observable characteristics
set Demogr := 1..nd;
set DS := 1..nd*ns;
set K2S := 1..nk2*ns;

set H := 1..nz ;         # index set of instrument including iv and X1
```

AMPL Model: MPEC_BLP.mod

```
## Define input data format:
param X1 {mj in MJ, k in K1} ;
param X2 {mj in MJ, k in K2} ;
param ActuShare {m in MJ} ;
param Z {mj in MJ, h in H} ;
param D {m in M, di in DS} ;
param v {m in M, k2i in K2S} ;
param invA {i in H, j in H} ;   # optimal weighting matrix = inv(Z'Z);
param OutShare {m in M} := 1 - sum {mj in (nbrn*(m-1)+1)..(nbrn*m)} ActuShare[mj];
```

AMPL Model: MPEC_BLP.mod

```
## Define variables

var theta1 {k in K1};
var SIGMA {k in K2};
var PI {k in K2, d in Demogr};
var delta {mj in MJ} ;

var EstShareIndivTop {mj in MJ, i in S} = exp( delta[mj]
+ sum {k in K2} (X2[mj,k]*SIGMA[k]*v[ceil(mj/nbrn), i+(k-1)*ns])
+ sum{k in K2, d in Demogr} (X2[mj,k]*PI[k,d]*D[ceil(mj/nbrn),i+(d-1)*ns]) );

var EstShareIndiv{mj in MJ, i in S} = EstShareIndivTop[mj,i] / (1+ sum{
l in ((ceil(mj/nbrn)-1)*nbrn+1)..(ceil(mj/nbrn)*nbrn)} EstShareIndivTop[l, i]);

var EstShare {mj in MJ} = 1/ns * (sum{i in S} EstShareIndiv[mj,i]) ;

var w {mj in MJ} = delta[mj] - sum {k in K1} (X1[mj,k]*theta1[k]) ;

var Zw {h in H} ;  ##  Zw{h in H} = sum {mj in MJ} Z[mj,h]*w[mj];
```

AMPL Model: MPEC_BLP.mod

```
minimize GMM : sum{h1 in H, h2 in H} Zw[h1]*invA[h1, h2]*Zw[h2];

subject to

     conZw {h in H}: Zw[h] = sum {mj in MJ} Z[mj,h]*w[mj] ;

     Shares {mj in MJ}: log(EstShare[mj]) = log(ActuShare[mj]) ;
```

AMPL/KNITRO Output

```
KNITRO 6.0.0: alg=1
opttol=1.0e-6
feastol=1.0e-6

Problem Characteristics
-----------------------
Objective goal:  Minimize
Number of variables:                    2338
    bounded below:                         0
    bounded above:                         0
    bounded below and above:               0
    fixed:                                 0
    free:                               2338
Number of constraints:                  2300
    linear equalities:                    44
    nonlinear equalities:               2256
    linear inequalities:                   0
    nonlinear inequalities:                0
    range:                                 0
Number of nonzeros in Jacobian:       131440
Number of nonzeros in Hessian:         58609
```

AMPL/KNITRO Output

```
  Iter      Objective      FeasError   OptError    ||Step||    CGits
--------  --------------  ----------  ----------  ----------  -------
       0    2.936110e+01   1.041e-04
       1    1.557550e+01   3.813e-01   4.561e-02   4.835e+01        9
       2    6.289721e+00   6.157e-01   2.605e+01   3.416e+02        0
       3    4.646499e+00   1.145e-01   3.041e+00   1.901e+02        0
       4    4.527042e+00   4.951e-02   5.887e-01   1.071e+02        0
       5    4.562016e+00   8.379e-03   4.865e-02   4.243e+01        0
       6    4.564521e+00   8.874e-05   6.051e-04   4.660e+00        0
       7    4.564553e+00   1.196e-08   6.356e-08   5.280e-02        0

EXIT: Locally optimal solution found.
```

AMPL/KNITRO Output

```
Final Statistics
----------------
Final objective value                  =   4.56455310841869e+00
Final feasibility error (abs / rel) =   1.20e-08 / 1.20e-08
Final optimality error  (abs / rel) =   6.36e-08 / 3.21e-09
# of iterations                        =          7
# of CG iterations                     =          9
# of function evaluations              =          8
# of gradient evaluations              =          8
# of Hessian evaluations               =          7
Total program time (secs)              =      10.48621 (    10.278 CPU time)
Time spent in evaluations (secs)       =       8.62244

===================================================
KNITRO 6.0.0: Locally optimal solution.
objective 4.564553108; feasibility error 1.2e-08
7 iterations; 8 function evaluations
```

The Example in Nevo (2000)

Method	NFXP
Software	Matlab
GMM Objective Value	14.90
# of θ steps	52
Function Evaluations	>1500
Timing	~ 24 sec.

The Example in Nevo (2000)

Method	MPEC	NFXP
Software	AMPL/IP	Matlab
GMM Objective Value	4.56	14.90^{*}
# of θ steps	7	52
Function Evaluations	8	$>$ 1500
Timing	$\sim$12 sec.	$\sim$ 24 sec.

The Example in Nevo (2000)

Method	MPEC	NFXP
Software	AMPL/IP	Matlab
GMM Objective Value	4.56	14.90^*
# of θ steps	7	52
Function Evaluations	8	$>$ 1500
Timing	$\sim$12 sec.	$\sim$ 24 sec.

$*$ Outer loop tolerance level $\epsilon_{out} = 0.1$. Coefficients on price and the interactions of price with demographic characteristics are one standard deviation away from the true solution

The Example in Nevo (2000)

	MPEC	NFXP	NFXP
Software	AMPL/IP	Matlab	Matlab
GMM Objective Value	4.56	14.90*	4.56**
# of θ steps	7	52	122
Function Evaluations	8	$>$ 1500	$>$ 2000
Timing	$\sim$12 sec.	$\sim$ 24 sec.	$\sim$66 sec.

$*$ Outer loop tolerance level $\epsilon_{out} = 0.1$. Coefficients on price and the interactions of price with demographic characteristics are one standard deviation away from the true solution

$**$ Outer loop tolerance level $\epsilon_{out} = 1.0\text{E}{-4}$; inner loop tolerance level $\epsilon_{in} = 1.0\text{E}{-12}$.

Monte Carlo in DFS11: Simulated Data Setup

- $$\begin{bmatrix} x_{1,j,t} \\ x_{2,j,t} \\ x_{3,j,t} \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -0.8 & 0.3 \\ -0.8 & 1 & 0.3 \\ 0.3 & 0.3 & 1 \end{bmatrix} \right)$$
- $\xi_{j,t} \sim N\left(0,1\right)$
- $p_{j,t} = \left|0.5 \cdot \xi_{j,t} + e_{j,t}\right| + 1.1 \cdot \left|\sum_{k=1}^{3} x_{k,jt,t}\right|$
- $z_{j,t,d} \sim N\left(\frac{1}{4} p_{j,t}, 1\right)$, $D = 6$ instruments
- $F_\beta\left(\beta;\theta\right)$: 5 independent normal distributions $\left(K = 3\right.$ attributes, price and the intercept)
- $\beta_i = \left\{\beta_i^0, \beta_i^1, \beta_i^2, \beta_i^3, \beta_i^p\right\}$: $E\left[\beta_i\right] = \left\{0.1, 1.5, 1.5, 0.5, -3\right\}$ and $\mathrm{Var}\left[\beta_\mathrm{i}\right] = \left\{0.5, 0.5, 0.5, 0.5, 0.2\right\}$

Implementation Details

- MATLAB, highly vectorized code, available at `http://faculty.chicagobooth.edu/jean-pierre.dube/research/MPECcode.html`
- Optimization software KNITRO
    - Professional quality optimization program
    - Can be called directly from R2008a version of MATLAB
    - We call from TOMLAB
- We provide sparsity pattern for $\nabla c(x)$ and $\nabla^2 \mathcal{L}(x)$ for MPEC
- We code exact first-order and second-order derivatives
    - Important for performance of smooth optimizers
    - With both 1st and 2nd derivatives, NFP is 3 to 10 times faster than using only 1st order derivatives
    - Same component functions for derivatives
    - Helpful for standard errors

Loose v.s. Tight Tolerances for NFXP

	NFXP Loose Inner	NFXP Loose Both	NFXP Tight	Truth
Fraction Convergence	0.0	0.54	0.95	
Frac.$<$ 1% $>$ "Global" Min.	0.0	0.0	1.00	
Mean Own Price Elasticity	-7.24	-7.49	-5.77	-5.68
Std. Dev. Own Price Elasticity	5.48	5.55	˜0	
Lowest Objective	0.0176	0.0198	0.0169	
Elasticity for Lowest Obj.	-5.76	-5.73	-5.77	-5.68

- 100 starting values for one dataset
- NFXP loose inner loop:$\epsilon_{\text{in}} = 10^{-4}$, $\epsilon_{\text{out}} = 10^{-6}$
- NFXP loose both: $\epsilon_{\text{in}} = 10^{-4}$, $\epsilon_{\text{out}} = 10^{-2}$
- NFXP tight: $\epsilon_{\text{in}} = 10^{-14}$, $\epsilon_{\text{out}} = 10^{-6}$

Lessons Learned

- Loose inner loop causes numerical error in gradient
  - Failure to diagonose convergence of outer loop
  - Leads to false estimates
- Making outer loop tolerance loose allows "convergence"
  - But to false solution

Speeds, # Convergences and Finite-Sample Performance

$T = 50, J = 25, nn = 1000$, 20 replications, 5 starting points/replication

Intercept $E\left[\beta_i^0\right]$	Lipsch. Const	Alg.	CPU (min)	Elasticities			Out. Share
				Bias	RMSE	Value	
-2	0.891	NFP	21.7	-0.077	0.14	-10.4	0.91
		MPEC	18.3	-0.076	0.14		
-1	0.928	NFP	28.3	-0.078	0.15	-10.5	0.86
		MPEC	16.3	-0.077	0.15		
0	0.955	NFP	41.7	-0.079	0.16	-10.6	0.79
		MPEC	15.2	-0.079	0.16		
1	0.974	NFP	71.7	-0.083	0.16	-10.7	0.69
		MPEC	11.8	-0.083	0.17		
2	0.986	NFP	103.3	-0.085	0.17	-10.8	0.58
		MPEC	13.5	-0.085	0.17		
3	0.993	NFP	166.7	-0.088	0.17	-11.0	0.46
		MPEC	10.7	-0.088	0.17		
4	0.997	NFP	300.0	-0.091	0.16	-11.0	0.35
		MPEC	12.7	-0.090	0.16		

of Function/Gradient/Hessian Evals and # Contraction Mapping Iterations

Intercept $E\left[\beta_i^0\right]$	Alg.	Func Eval	Grad/Hess Eval	Contraction Iter
-2	NFP	80	58	10,400
	MPEC	184	126	
-1	NFP	82	60	17,100
	MPEC	274	144	
0	NFP	77	56	29,200
	MPEC	195	113	
1	NFP	71	54	55,000
	MPEC	148	94	
2	NFP	68	50	84,000
	MPEC	188	107	
3	NFP	68	49	146,000
	MPEC	144	85	
4	NFP	81	50	262,000
	MPEC	158	100	

Lessons Learned

- For low Lipschitz constant, NFXP and MPEC about the same speed
- For high Lipschitz constant, NFXP becomes very slow
  - 1 hour per run for Intercept $= 4$
  - Reminder: you need to use more starting points if you want to find a good solution
- MPEC speed relatively invariant to Lipschitz constant
  - No contraction mapping in MPEC

Speed for Varying # of Markets, Products, Draws

T	J	nn	Lipsch. Const.	Alg	Runs	CPU (hr)	Outside Share
100	25	1000	0.999	NFP MPEC	80% 100%	10.9 0.3	0.45
250	25	1000	0.997	NFP MPEC	100% 100%	22.3 1.2	0.71
500	25	1000	0.998	NFP MPEC	80% 100%	65.6 2.5	0.65
100	25	3000	0.999	NFP MPEC	80% 100%	42.3 1.0	0.46
250	25	3000	0.997	NFP MPEC	100% 100%	80.0 3.0	0.71
25	100	1000	0.993	NFP MPEC	100% 100%	5.7 0.5	0.28
25	250	1000	0.999	NFP MPEC	100% 100%	28.4 2.3	0.07

of Function/Gradient/Hessian Evals and # Contraction Mapping Iterations

T	J	nn	Alg	# Iter.	Func. Eval.	Grad Eval.	Contrac. Mapping
100	25	1000	NFP MPEC	68 84	130 98	69 85	372,278
250	25	1000	NFP MPEC	58 118	82 172	59 119	246,000
500	25	1000	NFP MPEC	52 123	99 195	53 124	280,980
100	25	3000	NFP MPEC	60 83	171 114	61 84	479,578
250	25	3000	NFP MPEC	55 102	68 135	56 103	204,000
25	100	1000	NFP MPEC	54 97	71 145	55 98	198,114
25	250	1000	NFP MPEC	60 75	126 103	61 76	359,741

Summary

- Constrained optimization formulation for the random-coefficients demand estimation model is

$$\begin{array}{ll} \min\limits_{(\theta,\xi,g,)} & g^T W g \\ \text{subject to} & s(\delta;\theta_2) = S \\ & Z^T \xi = g \end{array}$$

- The constrained optimization approach (with good solvers) is reliable and has speed advantage
- It allows researchers to access best optimization solvers

Part IV

Estimation of Static Games

Structural Estimation of Games of Incomplete Information

- An active research topic in Applied Econometrics/Empirical Industrial Organization
  - Statis entry/exit games of incomplete information: Seim (2006), Su (2012)
  - Dynamic games of incomplete information: Aguirregabiria and Mira (2007), Bajari, Benkard, Levin (2007), Pakes, Ostrovsky, and Berry (2007), Pesendorfer and Schmidt-Dengler (2008), Pesendorfer and Schmidt-Dengler (2010), Kasahara and Shimotsu (2012), Egesdal, Lai and Su (2012) :
- Two main econometric issues appear in the estimation of these models
  - the existence of multiple equilibria – need to find all of them
  - computational burden in the solution of the game – repeated solving for equilibria for every guessed of structural parameters

Example: Static Game of Incomplete Information - due to John Rust

- Two firms: a and b
- Actions: each firm has two possible actions:

$$d_a = \begin{cases} 1 & \text{if firm } a \text{ choose to enter the market} \\ 0 & \text{if firm } a \text{ choose not to enter the market} \end{cases}$$

$$d_b = \begin{cases} 1 & \text{if firm } b \text{ choose to enter the market} \\ 0 & \text{if firm } b \text{ choose not to enter the market} \end{cases}$$

Example: Static Game of Incomplete Information

- Utility: Ex-post payoff to firms

$$u_a(d_a, d_b, x_a, \epsilon_a) = \begin{cases} [\alpha + d_b(\beta - \alpha)]\, x_a + \varepsilon_{a1}, & \text{if } d_a = 1, \\ 0 + \varepsilon_{a0}, & \text{if } d_a = 0; \end{cases}$$

$$u_b(d_a, d_b, x_b, \epsilon_b) = \begin{cases} [\alpha + d_a(\beta - \alpha)]\, x_b + \varepsilon_{b1}, & \text{if } d_b = 1, \\ 0 + \varepsilon_{b0}, & \text{if } d_b = 0; \end{cases}$$

  - (α, β): structural parameters to be estimated
  - (x_a, x_b): firms' observed types; **common knowledge**
  - $\varepsilon_a = (\varepsilon_{a0}, \varepsilon_{a1}), \varepsilon_b = (\varepsilon_{b0}, \varepsilon_{b1})$: firms' unobserved types, **private information**
  - $(\varepsilon_a, \varepsilon_b)$ are observed only by each firm, but not by their opponent firm nor by the econometrician

Example: Static Game of Incomplete Information

- Assume the error terms $(\varepsilon_a, \varepsilon_b)$ have a standardized type III extreme value distribution
- A Bayesian Nash equilibrium (p_a, p_b) satisfies

$$
\begin{aligned}
p_a &= \frac{\exp[p_b \beta x_a + (1 - p_b) \alpha x_a]}{1 + \exp[p_b \beta x_a + (1 - p_b) \alpha x_a]} \\
&= \frac{1}{1 + \exp[-x_a \alpha + p_b x_a (\alpha - \beta)]} \\
&\equiv \Psi_a(p_b, x_a; \alpha, \beta). \\
p_b &= \frac{1}{1 + \exp[-x_b \alpha + p_a x_b (\alpha - \beta)]} \\
&\equiv \Psi_b(p_a, x_b; \alpha, \beta).
\end{aligned}
$$

Static Game Example with One Market: Solving for Equilibria

- The true values of the structural parameters are

$$(\alpha, \beta) = (5, -11)$$

- There is only 1 market with observed types $(x_a, x_b) = (0.52, 0.22)$

$$\begin{aligned} p_a &= \frac{1}{1 + \exp\{0.52(-5) + p_b 0.52(16)\}} \\ p_b &= \frac{1}{1 + \exp\{0.22(-5) + p_a 0.22(16)\}} \end{aligned}$$

Static Game Example: Three Bayesian Nash Equilibria

Eq1: $(p_a, p_b) = (0.030100, 0.729886)$ stable under BR

Eq2: $(p_a, p_b) = (0.616162, 0.255615)$ unstable under BR

Eq3: $(p_a, p_b) = (0.773758, 0.164705)$ stable under BR

Static Game Example: Data Generation and Identification

- Data Generating Process (DGP): the data are generated by a single equilibrium
- The two players use the same equilibrium to play 1000 times
- Data: $X = \{(d_a^i, d_b^i)_{i=1}^{1000}, (x_a, x_b) = (0.52, 0.22)\}$
- Given data X, we want to recover structural parameters α and β

Static Game Example: Maximum Likelihood Estimation

- Maximize the likelihood function

$$
\begin{aligned}
\max_{(\alpha,\beta)} \quad & log\mathcal{L}\left(p_a(\alpha,\beta), p_b(\alpha,\beta); X\right) \\
& = \sum_{i=1}^{1000} \left(d_a^i * \log(p_a(\alpha,\beta)) + (1-d_a^i) * \log(1-p_a(\alpha,\beta))\right) \\
& + \sum_{i=1}^{1000} \left(d_b^i * \log(p_b(\alpha,\beta)) + (1-d_b^i) * \log(1-p_b(\alpha,\beta))\right)
\end{aligned}
$$

- $(p_a(\alpha,\beta), p_b(\alpha,\beta))$ are the solutions of the Bayesian-Nash Equilibrium equations

$$
\begin{aligned}
p_a &= \frac{1}{1+\exp\{-0.52(\alpha)+p_b 0.52(\alpha)-\beta\}} = \Psi_a(p_b, x_a, \alpha, \beta) \\
p_b &= \frac{1}{1+\exp\{-0.22(\alpha)+p_a 0.22(\alpha)-\beta\}} = \Psi_b(p_a, x_b, \alpha, \beta)
\end{aligned}
$$

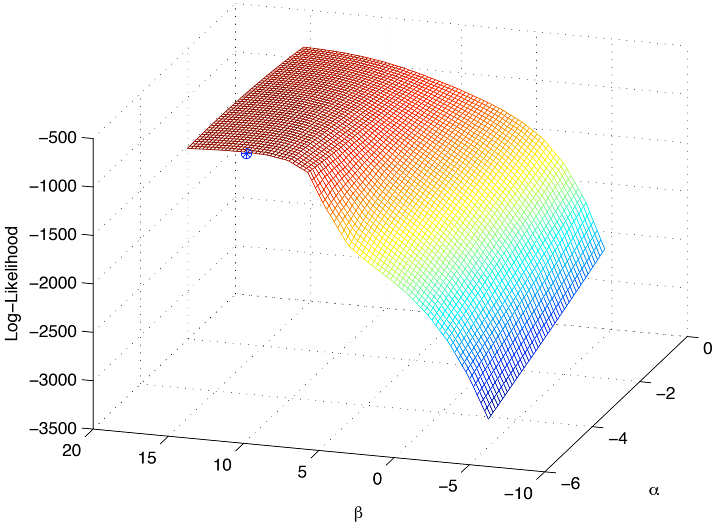
Static Game Example: MLE via NFXP

- Outer loop:
  - Choose (α, β) to maximize the likelihood function $log\mathcal{L}\left(p_a(\alpha, \beta), p_b(\alpha, \beta); X\right)$
- Inner loop:
  - For a given (α, β), solve the BNE equations for **ALL** equilibria: $(p_a^k(\alpha, \beta), p_b^k(\alpha, \beta)), \quad k = 1, \ldots, K$
  - Choose the equilibrium that gives the highest likelihood value:

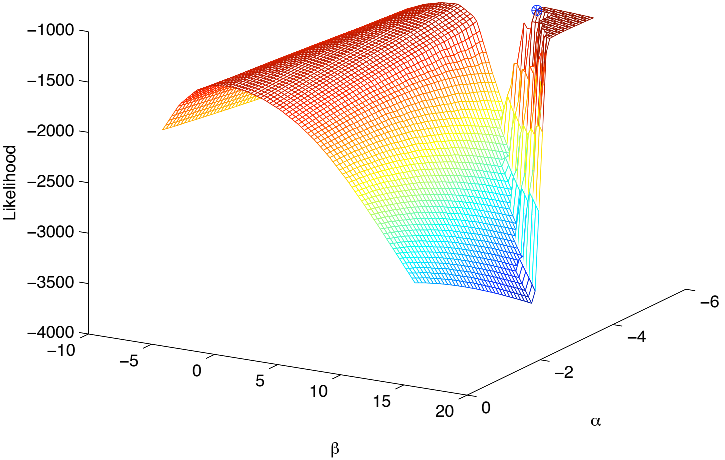
$$k^* = \underset{\{k=1,\ldots,K\}}{\text{argmax}} \; log\mathcal{L}\left(p_a^k(\alpha, \beta), p_b^k(\alpha, \beta); X\right)$$

$$(p_a(\alpha, \beta), p_b(\alpha, \beta)) = (p_a^{k^*}(\alpha, \beta), p_b^{k^*}(\alpha, \beta))$$

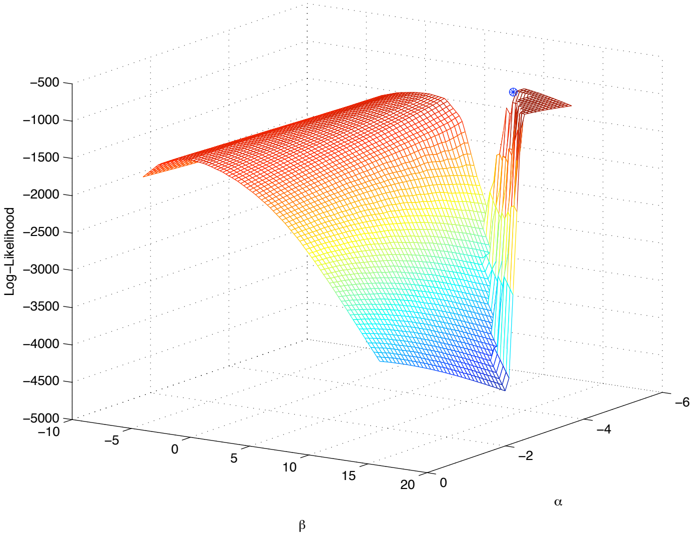
NFXP's Likelihood as a Function of (α, β) – Eq 1



NFXP's Likelihood as a Function of (α, β) – Eq 2



NFXP's Likelihood as a Function of (α, β) – Eq 3

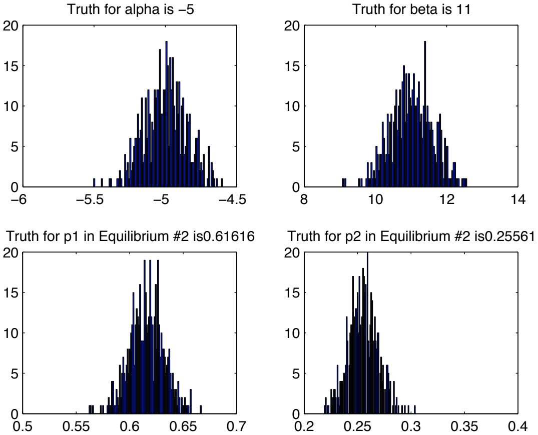


Constrained Optimization Formulation for Maximum Likelihood Estimation

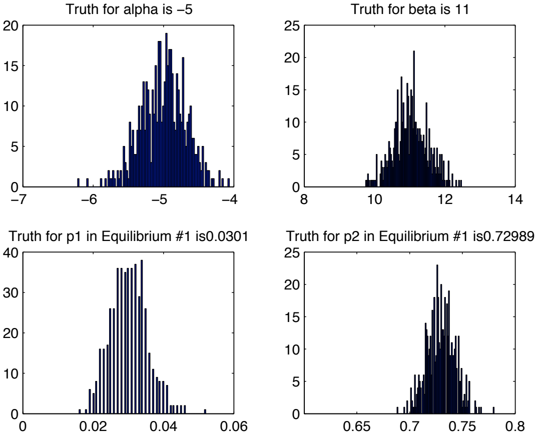
$$
\begin{aligned}
\max_{(\alpha,\beta,p_a,p_b)} \quad & log\mathcal{L}\,(p_a, p_b; X) \\
& = \sum_{i=1}^{1000} \left(d_a^i * \log(p_a) + (1 - d_a^i) * \log(1 - p_a) \right) \\
& + \sum_{i=1}^{1000} \left(d_b^i * \log(p_b) + (1 - d_b^i) * \log(1 - p_b) \right) \\
\text{subject to} \quad & p_a = \frac{1}{1 + \exp\{0.52(\alpha) + p_b 0.52(\alpha - \beta)\}} \\
& p_b = \frac{1}{1 + \exp\{-0.22(\alpha) + p_a 0.22(\alpha - \beta -)\}} \\
& 0 \le p_a, p_b \le 1
\end{aligned}
$$

Log-likelihood function is a smooth function of (p_a, p_b).

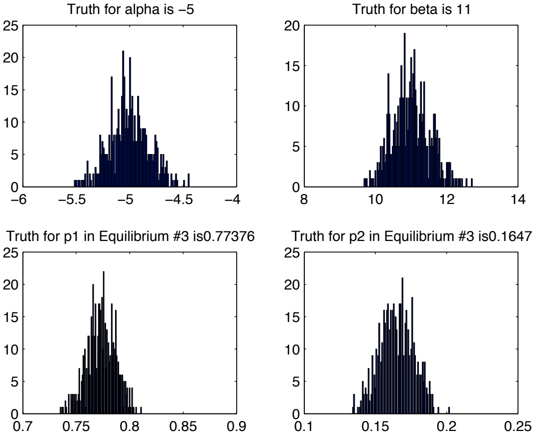
Monte Carlo Results with Eq2



Monte Carlo Results with Eq1



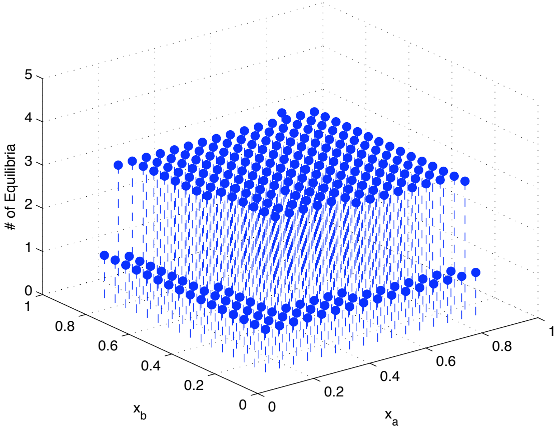
Monte Carlo Results with Eq3



Estimation with Multiple Markets

- There 256 different markets, i.e., 256 pairs of observed types (x_a^m, x_b^m), $m = 1, \ldots, 256$
- The grid on x_a has 16 points equally distributed between the interval $[0.12, 0.87]$, and similarly for x_b
- Use the same true parameter values: $(\alpha^0, \beta^0) = (-5, 11)$
- For each market with (x_a^m, x_b^m), solve BNE conditions for (p_a^m, p_b^m).
- There are multiple equilibria in most of 256 markets
- For each market, we (randomly) choose an equilibrium to generate 250 data points for that market
- The equilibrium used to generate data can be different in different markets

of Equilibria with Different (x_a^m, x_b^m)

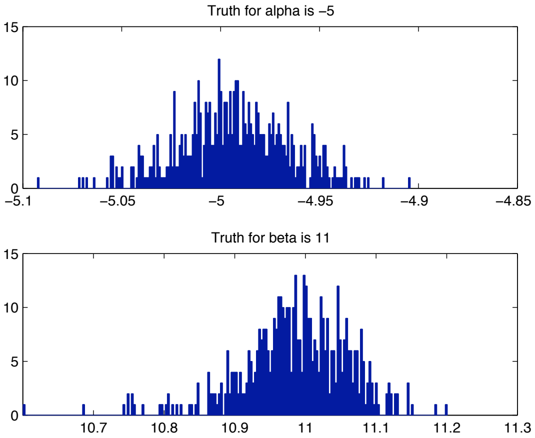


Estimation with Multiple Markets

- Constrained optimization formulation for MLE

$$
\begin{array}{cl}
\max\limits_{(\alpha,\beta,\{p_a^m,p_b^m\})} & \mathcal{L}\left(\{p_a^m, p_b^m\}, X\right) \\
\text{subject to} & p_a^m = \Psi_a(p_b^m, x_a^m, \alpha, \beta) \\
& p_b^m = \Psi_b(p_a^m, x_b^m, \alpha, \beta) \\
& 0 \le p_a^m, p_b^m \le 1, \quad m = 1, \ldots, 256.
\end{array}
$$

Static Game Example: Monte Carlo Results with Multiple Markets



2-Step Methods

- Recall the constrained optimization formulation for FIML is

$$\begin{array}{ll} \max\limits_{(\{\alpha,\beta,p_a,p_b\})} & \mathcal{L}\left(p_a, p_b, X\right) \\ \text{subject to} & p_a = \Psi_a(p_b, x_a, \alpha, \beta) \\ & p_b = \Psi_b(p_a, x_b, \alpha, \beta) \\ & 0 \le p_a, p_b \le 1 \end{array}$$

- Denote the solution as $(\alpha^*, \beta^*, p_a^*, p_b^*)$
- Suppose we know (p_a^*, p_b^*), how do we recover (α^*, β^*)?

2-Step Methods: Recovering (α^*, β^*)

- Idea 1: Solve the BNE equations for (α^*, β^*):

$$\begin{aligned} p_a^* &= \Psi_a(p_b^*, x_a, \alpha, \beta) \\ p_b^* &= \Psi_b(p_a^*, x_b, \alpha, \beta) \end{aligned}$$

- Idea 2: Choose (α, β) to

$$\max_{(\alpha,\beta)} \mathcal{L}\left(\Psi_a(p_b^*, x_a, \alpha, \beta), \Psi_b(p_a^*, x_b, \alpha, \beta), X\right)$$

2-Step Methods

- Idea 1
  - Step 1: Estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$ from the data
  - Step 2: Solve

$$\begin{array}{rcl} \hat{p_a} & = & \Psi_a(\hat{p_b}, x_b, \alpha, \beta) \\ \hat{p_b} & = & \Psi_b(\hat{p_b}, x_b, \alpha, \beta) \end{array}$$

- Idea 2
  - Step 1: Estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$ from the data
  - Step 2:

$$\max_{(\alpha,\beta)} \mathcal{L}\left(\Psi_a(\hat{p_b}, x_a, \alpha, \beta), \Psi_b(\hat{p_a}, x_b, \alpha, \beta), X\right)$$

2-Step Methods: Potential Issues to be Addressed

- How do we estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$?
  - Different methods give different $\hat{p}$
  - One method is the frequency estimator:

$$\hat{p_a} = \frac{1}{N} \sum_{i=1}^{N} I_{\{d_a^i = 1\}}$$
$$\hat{p_b} = \frac{1}{N} \sum_{i=1}^{N} I_{\{d_b^i = 1\}}$$

- If $(\hat{p_a}, \hat{p_b}) \neq (p_a^*, p_b^*)$, then $(\hat{\alpha}, \hat{\beta}) \neq (\alpha^*, \beta^*)$
- For a given $(\hat{p_a}, \hat{p_b})$, there might not be a solution to the BNE equations

$$\begin{aligned} \hat{p_a} &= \Psi_a(\hat{p_b}, x_a, \alpha, \beta) \\ \hat{p_b} &= \Psi_b(\hat{p_a}, x_b, \alpha, \beta) \end{aligned}$$

2-Step Methods: Pseudo Maximum Likelihood

- In 2-tep methods
  - Step 1: Estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$
  - Step 2: Solve

$$
\begin{array}{ll}
\max\limits_{(\{\alpha,\beta,p_a,p_b\})} & \mathcal{L}\left(p_a, p_b, X\right) \\
\text{subject to} & p_a = \Psi_a(\hat{p_b}, x_a, \alpha, \beta) \\
& p_b = \Psi_b(\hat{p_a}, x_b, \alpha, \beta) \\
& 0 \le p_a, p_b \le 1
\end{array}
$$

- Or equivalently
  - Step 1: Estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$
  - Step 2: Solve

$$
\max_{(\{\alpha,\beta\})} \quad \mathcal{L}\left(\Psi_a(\hat{p_b}, x_a, \alpha, \beta), \Psi_b(\hat{p_a}, x_b, \alpha, \beta), X\right)
$$

2-Step Methods: Least Square Estimators

- Pesendofer and Schmidt-Dengler (2008)
  - Step 1: Estimate $\hat{p} = (\hat{p_a}, \hat{p_b})$ from the data
  - Step 2:

$$\min_{(\alpha,\beta)} \quad \left\{ (\hat{p_a} - \Psi_a(\hat{p_b}, x_a, \alpha, \beta))^2 + (\hat{p_b} - \Psi_b(\hat{p_b}, x_b, \alpha, \beta))^2 \right\}$$

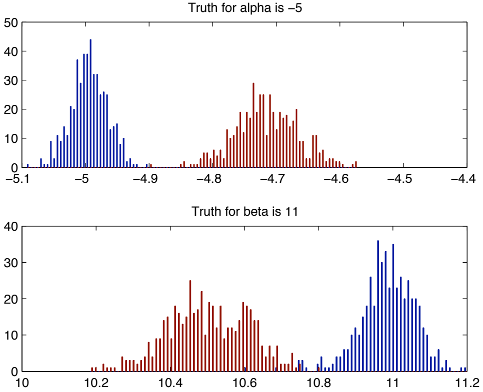
- For dynamic games, Markov perfect equilibrium conditions are characterized by

$$p = \Psi(p, \theta)$$

- Step 1: Estimate $\hat{p}$ from the data
- Step 2:

$$\min_{\theta} \quad [\hat{p} - \Psi(\hat{p}, \theta)]' W [\hat{p} - \Psi(\hat{p}, \theta)]'$$

Static Game Example: ML v.s. 2-Step PML



- Pakes, Ostrovsky, and Berry (2007): pseudo likelihood function is not a suitable criterion function in a 2-step estimator and can lead to large bias.

Nested Pseudo Likelihood (NPL): Aguirregabiria and Mira (2007)

- NPL iterates on the 2-step methods

  1. Estimate $\hat{p}^0 = (\hat{p}_a^0, \hat{p}_b^0)$, set $k = 0$
  2. **REPEAT**

     2.1 Solve

     $$(\alpha^{k+1}, \beta^{k+1}) = \arg\max_{(\alpha,\beta)} \mathcal{L}\left(\Psi_a(\hat{p}_b^k, x_a, \alpha, \beta), \Psi_b(\hat{p}_a^k, x_b, \alpha, \beta), X\right)$$

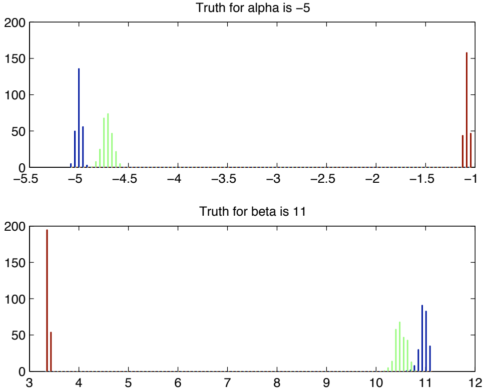
     2.2 One best-reply iteration on $\hat{p}^k$

     $$\begin{array}{rcl} \hat{p}_a^{k+1} & = & \Psi_a(\hat{p}_b^k, x_a, \alpha^{k+1}, \beta^{k+1}) \\ \hat{p}_b^{k+1} & = & \Psi_b(\hat{p}_a^k, x_b, \alpha^{k+1}, \beta^{k+1}) \end{array}$$

     2.3 Let $k := k + 1$;

     UNTIL convergence in (α^k, β^k) and $(\hat{p}_a^k, \hat{p}_b^k)$

Static Game Example: ML, 2-Step PML and NPL



- Some equilibria in the data are NOT best-reply (Lyapunov) stable
- Pesendofer (2010): Best-reply stable is not a reasonable equilibrium selection rule in games of incomplete information

Design of Data Generating Process in Monte Carlo Experiments

- Monte Carlo 1: Randomly selected equilibrium in each market
  - In each market, we randomly choose an equilibrium to generate data
- Monte Carlo 2: Best-response stable equilibrium with lowest probabilities of entering for firm a
  - In each market, we choose the equilibrium that results in the lower probability of entering for firm a to generate data. These equilibria are stable under Best-Reply iteration.
- Monte Carlo 3: Best-response stable equilibrium with lowest probabilities of entering for firm a
  - In each market, we randomly choose an equilibrium that is stable under Best-Reply iteration to generate data.
- In each experiment, we vary the number of repeated observations T. For each experiment and each T, we generate 100 datasets and estimate the model using each of the 100 datasets

Monte Carlo 1: Random Equilibrium in Each Market

- In each market, we randomly choose an equilibrium to generate data

Monte Carlo 1: $T = 5$ and 10 for Each Market

T	Estimator	Estimates α	Estimates β	RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
	Truth	5	-11	–	–	–	–
5	ML (Cons. Opt.)	5.027 (0.179)	-10.743 (0.585)	0.661	1.346	100	–
5	2-Step PML	3.068 (0.208)	-7.279 (0.512)	4.228	0.043	100	–
5	2-Step LS	2.918 (0.203)	-7.597 (0.654)	4.047	0.048	100	–
5	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	31.527	0	1000
10	ML (Cons. Opt.)	5.029 (0.126)	-10.816 (0.326)	0.394	0.641	100	–
10	2-Step PML	3.719 (0.165)	-8.535 (0.403)	2.812	0.042	100	–
10	2-Step LS	3.459 (0.164)	-8.499 (0.531)	2.990	0.049	100	–
10	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	35.756	0	1000

Monte Carlo 1: $T = 25$ and 50 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
25	ML (Cons. Opt.)	5.018 (0.084)	-10.964 (0.166)	0.189	0.512	100	–
25	2-Step PML	4.302 (0.122)	-9.663 (0.268)	1.537	0.060	100	–
25	2-Step LS	3.959 (0.134)	-9.311 (0.354)	2.019	0.050	100	–
25	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	52.268	0	1000
50	ML (Cons. Opt.)	5.005 (0.056)	-11.007 (0.139)	0.150	0.669	100	–
50	2-Step PML	4.590 (0.099)	-10.280 (0.230)	0.865	0.093	100	–
50	2-Step LS	4.279 (0.109)	-9.895 (0.283)	1.354	0.052	100	–
50	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	82.390	0	1000

Monte Carlo 1: $T = 100$ and 250 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
100	ML (Cons. Opt.)	5.006 (0.045)	-10.997 (0.092)	0.102	1.252	100	–
100	2-Step PML	4.773 (0.067)	-10.607 (0.165)	0.487	0.174	100	–
100	2-Step LS	4.533 (0.084)	-10.285 (0.200)	0.881	0.053	100	–
100	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	150.220	0	1000
250	ML (Cons. Opt.)	5.000 (0.028)	-10.999 (0.057)	0.063	2.512	100	–
250	2-Step PML	4.905 (0.043)	-10.828 (0.114)	0.231	0.410	100	–
250	2-Step LS	4.905 (0.051)	-10.624 (0.157)	0.472	0.054	100	–
250	NPL (freq. prob.)	N/A (N/A)	N/A (N/A)	N/A	351.990	0	1000

Monte Carlo 2: B-R Stable Equilibrium with Lowest Probabilities of Entering for Firm a

- In each market, we choose the equilibrium that results in the lower probability of entering for firm a to generate data
- These equilibria are stable under Best-Reply iteration.

Monte Carlo 2: $T = 5$ and 10 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
5	ML (Cons. Opt.)	5.234 (0.278)	-11.238 (0.506)	0.665	0.692	100	–
5	2-Step PML	4.459 (0.276)	-10.646 (0.796)	1.058	0.040	100	–
5	2-Step LS	4.514 (0.347)	-11.369 (1.100)	1.300	0.053	100	–
5	NPL (freq. prob.)	4.863 (0.241)	-10.019 (1.830)	1.639	36.051	2	987
10	ML (Cons. Opt.)	5.065 (0.143)	-11.111 (0.345)	0.393	0.441	100	–
10	2-Step PML	4.787 (0.165)	-10.886 (0.529)	0.602	0.043	100	–
10	2-Step LS	4.914 (0.238)	-11.473 (0.852)	1.002	0.055	100	–
10	NPL (freq. prob.)	5.054 (0.241)	-10.411 (1.830)	0.958	33.153	29	808

Monte Carlo 2: $T = 25$ and 50 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
25	ML (Cons. Opt.)	5.018 (0.076)	-11.022 (0.181)	0.197	0.417	100	–
25	2-Step PML	4.926 (0.114)	-11.040 (0.264)	0.298	0.058	100	–
25	2-Step LS	5.014 (0.147)	-11.387 (0.479)	0.632	0.057	100	–
25	NPL (freq. prob.)	4.995 (0.081)	-10.607 (0.563)	0.688	29.122	71	543
50	ML (Cons. Opt.)	5.000 (0.061)	-11.000 (0.133)	0.146	0.398	100	–
50	2-Step PML	4.956 (0.080)	-10.983 (0.198)	0.218	0.090	100	–
50	2-Step LS	5.007 (0.109)	-11.119 (0.329)	0.365	0.056	100	–
50	NPL (freq. prob.)	4.998 (0.070)	-10.665 (0.472)	0.581	32.133	86	409

Monte Carlo 2: $T = 100$ and 250 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
100	ML (Cons. Opt.)	5.005 (0.046)	-10.996 (0.103)	0.112	0.858	100	–
100	2-Step PML	4.985 (0.060)	-11.011 (0.164)	0.175	0.164	100	–
100	2-Step LS	5.011 (0.077)	-11.090 (0.238)	0.265	0.056	100	–
100	NPL (freq. prob.)	5.005 (0.051)	-10.908 (0.283)	0.301	34.516	96	242
250	ML (Cons. Opt.)	5.000 (0.031)	-10.995 (0.062)	0.069	1.798	100	–
250	2-Step PML	4.994 (0.037)	-11.002 (0.092)	0.099	0.379	100	–
250	2-Step LS	5.005 (0.042)	-11.025 (0.152)	0.160	0.057	100	–
250	NPL (freq. prob.)	5.002 (0.051)	-10.955 (0.283)	0.198	57.083	100	174

Monte Carlo 3: B-R Stable Equilibrium in Each Market

- In each market, we randomly choose an equilibrium that is stable under Best-Reply iteration to generate data.

Monte Carlo 3: $T = 5$ and 10 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
5	ML (Cons. Opt.)	5.197 (0.245)	-11.189 (0.463)	0.588	0.803	100	–
5	2-Step PML	4.380 (0.263)	-10.427 (0.711)	1.132	0.040	100	–
5	2-Step LS	4.395 (0.318)	-11.131 (1.078)	1.278	0.053	100	–
5	NPL (freq. prob.)	4.707 (0.241)	-8.534 (1.830)	2.574	34.847	4	975
10	ML (Cons. Opt.)	5.104 (0.149)	-11.038 (0.304)	0.354	0.472	100	–
10	2-Step PML	4.787 (0.181)	-10.831 (0.523)	0.615	0.043	100	–
10	2-Step LS	4.893 (0.243)	-11.418 (0.805)	0.942	0.055	100	–
10	NPL (freq. prob.)	5.019 (0.148)	-9.732 (0.753)	1.534	28.135	46	682

Monte Carlo 3: $T = 25$ and 50 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
25	ML (Cons. Opt.)	5.040 (0.085)	-10.992 (0.193)	0.214	0.245	100	–
25	2-Step PML	4.945 (0.113)	-10.943 (0.307)	0.335	0.059	100	–
25	2-Step LS	5.022 (0.135)	-11.175 (0.509)	0.553	0.057	100	–
25	NPL (freq. prob.)	5.032 (0.088)	-10.087 (0.824)	1.229	25.661	75	469
50	ML (Cons. Opt.)	5.009 (0.060)	-10.999 (0.149)	0.160	0.380	100	–
50	2-Step PML	4.967 (0.089)	-10.990 (0.203)	0.223	0.091	100	–
50	2-Step LS	5.016 (0.111)	-11.106 (0.343)	0.374	0.058	100	–
50	NPL (freq. prob.)	5.018 (0.071)	-10.243 (0.780)	1.087	30.148	86	384

Monte Carlo 3: $T = 100$ and 250 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
100	ML (Cons. Opt.)	5.011 (0.046)	-10.982 (0.095)	0.107	0.821	100	–
100	2-Step PML	4.995 (0.060)	-11.011 (0.164)	0.176	0.164	100	–
100	2-Step LS	5.022 (0.077)	-11.090 (0.249)	0.275	0.059	100	–
100	NPL (freq. prob.)	5.024 (0.060)	-10.661 (0.650)	0.733	30.406	99	225
250	ML (Cons. Opt.)	5.003 (0.025)	-10.993 (0.057)	0.062	1.838	100	–
250	2-Step PML	4.9957 (0.034)	-11.000 (0.103)	0.108	0.377	100	–
250	2-Step LS	5.008 (0.040)	-11.025 (0.171)	0.176	0.060	100	–
250	NPL (freq. prob.)	5.010 (0.060)	-10.854 (0.650)	0.470	53.572	100	168

Monte Carlo 3: $T = 100$ and 250 for Each Market

T	Estimator	Estimates		RMSE	CPU (sec.)	# of Data Sets	Avg. NPL Iter.
		α	β				
	Truth	5	-11	–	–	–	–
100	ML (Cons. Opt.)	5.011 (0.046)	-10.982 (0.095)	0.107	0.821	100	–
100	2-Step PML	4.995 (0.060)	-11.011 (0.164)	0.176	0.164	100	–
100	2-Step LS	5.022 (0.077)	-11.090 (0.249)	0.275	0.059	100	–
100	NPL (freq. prob.)	5.024 (0.060)	-10.661 (0.650)	0.733	30.406	99	225
250	ML (Cons. Opt.)	5.003 (0.025)	-10.993 (0.057)	0.062	1.838	100	–
250	2-Step PML	4.9957 (0.034)	-11.000 (0.103)	0.108	0.377	100	–
250	2-Step LS	5.008 (0.040)	-11.025 (0.171)	0.176	0.060	100	–
250	NPL (freq. prob.)	5.010 (0.060)	-10.854 (0.650)	0.470	53.572	100	168

Dynamic Game: Egesdal, Lai and Su (2012)

The Example in Kasahara and Shimotsu (2012) with $\theta_{RN} = 2$.

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

Dynamic Game: Egesdal, Lai and Su (2012)

The Example in Kasahara and Shimotsu (2012) with $\theta_{RN} = 4$.

M	T	Estimator	Estimates		CPU Time Per Run (sec.)	# of Data Sets Converged	Avg. NPL(-Λ) Iter.
			θ_{RN}	θ_{RS}			
		Truth	4	1	–	–	–
400	1	MLE	4.055 (0.613)	1.003 (0.158)	0.61	100	–
400	1	2S-PML	3.107 (0.442)	0.839 (0.099)	0.02	100	–
400	1	NPL	N/A (N/A)	N/A (N/A)	1.68	0	100
400	1	Λ-NPL	N/A (N/A)	N/A (N/A)	1.68	0	100
400	10	MLE	4.003 (0.039)	1.000 (0.016)	0.50	100	–
400	10	2S-PML	3.902 (0.099)	0.983 (0.025)	0.04	100	–
400	10	NPL	N/A (N/A)	N/A (N/A)	7.61	0	250
400	10	Λ-NPL	N/A (N/A)	N/A (N/A)	7.54	0	250
400	20	MLE	4.003 (0.032)	1.001 (0.011)	0.47	100	–
400	20	2S-PML	3.954 (0.084)	0.992 (0.019)	0.06	100	–
400	20	NPL	N/A (N/A)	N/A (N/A)	12.38	0	250
400	20	Λ-NPL	N/A (N/A)	N/A (N/A)	12.41	0	250

Conclusion

- NPL (Aguirregabiria and Mira 2007) is not an appropriate method for estimating games
- Estimation of dynamic games is an interesting but challenging computational optimization problem
  - Exploring sparsity patterns in constraint Jacobian and Hessian in numerical implementation
- Ongoing research
  - Estimation of dynamic discrete choice games of incomplete information – Egesdal, Lai and Su (2012)

Part V

Estimation of Dynamic Games

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	?	?	?	?	?
2						
3						
4						
5						
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2						
3						
4						
5						
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	?	?	?	?	?
3						
4						
5						
6						
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3						
4						
5						
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \dots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	?	?	?	?	?
4						
5						
6						
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	0	1	0	0	1
4						
5						
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	0	1	0	0	1
4	5	?	?	?	?	?
5						
6						
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \dots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	0	1	0	0	1
4	5	1	1	0	0	0
5						
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	0	1	0	0	1
4	5	1	1	0	0	0
5	5	1	1	0	0	0
6						
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Extry/Exit Games: An Illustrating Example

- Five firms: $i = 1, \ldots, 5$
- Firm i's decision in period t:

$$a_i^t = 0\text{: exit (inactive);} \quad a_i^t = 1\text{: enter (active)}$$

- Simultaneous decisions conditional on observing the market size, all firms' decisions in the last period and private shocks

Time	Market Size	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
0	2	0	0	0	0	0
1	3	0	1	0	0	1
2	4	0	1	0	1	1
3	5	0	1	0	0	1
4	5	1	1	0	0	0
5	5	1	1	0	0	1
6	6	1	1	1	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Estimation Methods for Discrete-Choice Games of Incomplete Information

- Maximum-Likelihood (ML) estimator
  - Efficient estimator in large-sample theory
  - Expensive to compute
- Two-step estimators: Bajari, Benkard, Levin (2007), Pesendorfer and Schmidt-Dengler (2008), Pakes, Ostrovsky, and Berry (2007)
  - Computationally simple
  - Potentially large finite-sample biases
  - Loss of efficiency in large-sample theory
- Nested Pseudo Likelihood (NPL) estimator: Aguirregabiria and Mira (2007), Kasahara and Shimotsu (2012)

What We Do in This Paper: Egesdal, Lai and Su (2012)

- Propose a constrained optimization formulation for the ML estimator to estimate dynamic games
- Conduct Monte Carlo experiments to compare performance of different estimators
  - Two-step pseudo maximum likelihood (2S-PML) estimator
  - NPL estimator implemented by NPL algorithm and NPL-Λ algorithm
  - ML estimator via the constrained optimization approach

The Dynamic Game Model in AM (2007)

- Discrete time infinite-horizon: $t = 1, 2, ..., \infty$
- N players: $i \in \mathcal{I} = \{1, ..., N\}$
- The market is characterized by size $s^t \in \mathcal{S} = \{s_1, \ldots, s_L\}$.
  - market size is observed by all players
  - exogenous and stationary market size transition: $f_{\mathcal{S}}(s^{t+1}|s^t)$
- At the beginning of each period t, player i observes $(\boldsymbol{x}^t, \boldsymbol{\varepsilon}_i^t)$
  - $\boldsymbol{x}^t$: a vector of common-knowledge state variables
  - $\boldsymbol{\varepsilon}_i^t$: private shocks
- Players then simultaneously choose whether to be active in the market in that period
  - $a_i^t \in \mathcal{A} = \{0, 1\}$: player i's action in period t
  - $\boldsymbol{a}^t = (a_1^t, \ldots, a_N^t)$: the collection of all players' actions.
  - $\boldsymbol{a}_{-i}^t = (a_1^t, \ldots, a_{i-1}^t, a_{i+1}^t, \ldots, a_N^t)$: the current actions of all players other than i

State Variables

- Common-knowledge state variables: $\boldsymbol{x}^t = \left(s^t, \boldsymbol{a}^{t-1}\right)$
- Private shocks: $\boldsymbol{\varepsilon}_i^t = \left\{\varepsilon_i^t\left(a_i^t\right)\right\}_{a_i^t \in \mathcal{A}}$
  - $\varepsilon_i^t\left(a_i^t\right)$ has a i.i.d type-I extreme value distribution across actions and players as well as over time
  - opposing players know only its probability density function $g(\boldsymbol{\varepsilon}_i^t)$.
- The **conditional independence** assumption on state transition:

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

Player i's Utility Maximization Problem

- $\boldsymbol{\theta}$: the vector of structural parameters
- $\beta \in (0, 1)$: the discount factor.
- player i's per-period payoff function:

$$\tilde{\Pi}_i \left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t, \boldsymbol{\varepsilon}_i^t; \boldsymbol{\theta}\right) = \Pi_i \left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t; \boldsymbol{\theta}\right) + \varepsilon_i^t \left(a_i^t\right)$$

- The common-knowledge component of the per-period payoff

$$\Pi_i \left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t; \boldsymbol{\theta}\right)$$

$$= \begin{cases} \theta^{RS} s^t - \theta^{RN} \log \left(1 + \sum_{j \neq i} a_j^t\right) - \theta_i^{FC} - \theta^{EC} \left(1 - a_i^{t-1}\right), & \text{if } a_i^t = 1, \\ 0 & \text{if } a_i^t = 0, \end{cases}$$

- Player i's utility maximization problem:

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

Equilibrium Concept: Markov Perfect Equilibrium

- Equilibrium characterization in terms of the observed states $\boldsymbol{x}$
- $P_i(a_i|\boldsymbol{x})$: the conditional choice probability of player i choosing action a_i at state $\boldsymbol{x}$
- $V_i(\boldsymbol{x})$: the expected value function for player i at state $\boldsymbol{x}$
- Define $\boldsymbol{P} = \{P_i(a_i|\boldsymbol{x})\}_{i\in\mathcal{I},a_i\in\mathcal{A},\boldsymbol{x}\in\mathcal{X}}$ and $\boldsymbol{V} = \{V_i(\boldsymbol{x})\}_{i\in\mathcal{I},\boldsymbol{x}\in\mathcal{X}}$
- A Markov perfect equilibrium is a vector $(\boldsymbol{V}, \boldsymbol{P})$ that satisfies two systems of nonlinear equations:
  - Bellman equation (for each player i)
  - Bayes-Nash equilibrium conditions

System I: Bellman Optimality

- **Bellman Optimality**. $\forall i \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}$

$$V_i(\boldsymbol{x}) = \sum_{a_i \in \mathcal{A}} P_i(a_i|\boldsymbol{x}) \left[\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta}) + e_i^{\boldsymbol{P}}(a_i, \boldsymbol{x})\right] + \beta \sum_{\boldsymbol{x}' \in \mathcal{X}} V_i(\boldsymbol{x}') f_{\mathcal{X}}^{\boldsymbol{P}}(\boldsymbol{x}'|\boldsymbol{x})$$

- $\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta})$: the expected payoff of $\Pi_i(a_i, \boldsymbol{a}_{-i}, \boldsymbol{x}; \boldsymbol{\theta})$ for player i from choosing action a_i at state $\boldsymbol{x}$ and given $P_j(a_j|\boldsymbol{x})$,

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

- $f_{\mathcal{X}}^{\boldsymbol{P}}(\boldsymbol{x}'|\boldsymbol{x})$: state transition probability of $\boldsymbol{x}$, given $\boldsymbol{P}$

$$f_{\mathcal{X}}^{\boldsymbol{P}}[\boldsymbol{x}' = (s', \boldsymbol{a}')|\boldsymbol{x} = (s, \tilde{\boldsymbol{a}})] = \left[\prod_{j=1}^{N} P_j(a'_j|\boldsymbol{x})\right] f_{\mathcal{S}}(s'|s)$$

- $$e_i^{\boldsymbol{P}}(a_i, \boldsymbol{x}) = \text{Euler's Constant} - \sigma \log\left[P_i(a_i|\boldsymbol{x})\right]$$

System II: Bayes-Nash Equilibrium Conditions

- **Bayes-Nash Equilibrium**.

$$P_i\left(a_i = j|\boldsymbol{x}\right) = \frac{\exp\left[v_i\left(a_i = j|\boldsymbol{x}\right)\right]}{\sum\limits_{k\in\mathcal{A}}\exp\left[v_i\left(a_i = k|\boldsymbol{x}\right)\right]}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{A}, \boldsymbol{x} \in \mathcal{X},$$

- $v_i\left(a_i|\boldsymbol{x}\right)$: choice-specific expected value function

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

- $f_i^{\boldsymbol{P}}\left(\boldsymbol{x}'|\boldsymbol{x},a_i\right)$: the state transition probability conditional on the current state $\boldsymbol{x}$, player i's action a_i, and his beliefs $\boldsymbol{P}$

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

Markov Perfect Equilibrium

- **Bellman Optimality**. $\forall i \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}$

$$V_i(\boldsymbol{x}) = \sum_{a_i \in \mathcal{A}} P_i(a_i|\boldsymbol{x}) \left[\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta}) + e_i^{\boldsymbol{P}}(a_i, \boldsymbol{x})\right] + \beta \sum_{\boldsymbol{x}' \in \mathcal{X}} V_i(\boldsymbol{x}') f_{\mathcal{X}}^{\boldsymbol{P}}(\boldsymbol{x}'|\boldsymbol{x})$$

- **Bayes-Nash Equilibrium**.

$$P_i(a_i = j|\boldsymbol{x}) = \frac{\exp\left[v_i(a_i = j|\boldsymbol{x})\right]}{\sum_{k \in \mathcal{A}} \exp\left[v_i(a_i = k|\boldsymbol{x})\right]}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{A}, \boldsymbol{x} \in \mathcal{X},$$

- In compact notation

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

- Set of all Markov Perfect Equilibria

$$SOL(\Psi, \boldsymbol{\theta}) = \left\{ (\boldsymbol{P}, \boldsymbol{V}) \middle| \begin{array}{rcl} \boldsymbol{V} &=& \Psi^{\boldsymbol{V}}(\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\theta}) \\ \boldsymbol{P} &=& \Psi^{\boldsymbol{P}}(\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\theta}) \end{array} \right\}$$

Data Generating Process

- $\boldsymbol{\theta}^0$: the true value of structural parameters in the population
- $(\boldsymbol{V}^0, \boldsymbol{P}^0)$: a Markov perfect equilibrium at $\boldsymbol{\theta}^0$
- **Assumption:** If multiple Markov perfect equilibria exist, only one equilibrium is played in the data
- Data: $\boldsymbol{Z} = \left\{\bar{\boldsymbol{a}}^{mt}, \bar{\boldsymbol{x}}^{mt}\right\}_{m\in\mathcal{M}, t\in\mathcal{T}}$
    - observations from M independent markets over T periods
    - In each market m and time period t, researchers observe
        - the common-knowledge state variables $\bar{\boldsymbol{x}}^{mt}$
        - players' actions $\bar{\boldsymbol{a}}^{mt} = (\bar{a}_1^{mt}, \ldots, \bar{a}_N^{mt})$

Maximum-Likelihood Estimation

- For a given $\boldsymbol{\theta}$, let $\left(\boldsymbol{P}^{\ell}(\boldsymbol{\theta}), \boldsymbol{V}^{\ell}(\boldsymbol{\theta})\right) \in SOL(\Psi, \boldsymbol{\theta})$ be the ℓ-th equilibrium
- Given data $\boldsymbol{Z} = \left\{\bar{\boldsymbol{a}}^{mt}, \bar{\boldsymbol{x}}^{mt}\right\}_{m \in \mathcal{M}, t \in \mathcal{T}}$, the logarithm of the likelihood function is

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

- The ML estimator is

$$\boldsymbol{\theta}^{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ L(Z, \boldsymbol{\theta}) \tag{1}$$

ML Estimation via Constrained Optimization Approach

- Given data $\boldsymbol{Z} = \left\{ \bar{\boldsymbol{a}}^{mt}, \bar{\boldsymbol{x}}^{mt} \right\}_{m \in \mathcal{M}, t \in \mathcal{T}}$, the logarithm of the augmented likelihood function is

$$\mathcal{L}\left(\boldsymbol{Z}, \boldsymbol{P}\right) = \frac{1}{M} \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{t=1}^{T} \log P_i\left(\bar{a}_i^{mt} | \bar{\boldsymbol{x}}^{mt}\right).$$

- The constrained optimization formulation of the ML estimation problem is

$$\begin{array}{cl} \max\limits_{(\boldsymbol{\theta}, \boldsymbol{P}, \boldsymbol{V})} & \mathcal{L}\left(\boldsymbol{Z}, \boldsymbol{P}\right) \\ \text{subject to} & \boldsymbol{V} = \Psi^{\boldsymbol{V}}\left(\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\theta}\right) \\ & \boldsymbol{P} = \Psi^{\boldsymbol{P}}\left(\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\theta}\right) \end{array} \qquad (2)$$

- **Thm.** Problem (1) and (2) have the same solution.

AMPL Code

- AMPL files:
  - AMPL Model File: `estimate_constr.mod`
  - AMPL Data File: `estimate_Used.dat`
  - AMPL Command File: `estimate_constr.run`
  - Remember to change the path to the KNITRO (or other) solver on your computer
- Solve an optimization problem in AMPL:
  - ampl:
    ampl:
    ampl: include estimate_constr.run

AMPL/KNITRO Output

```
KNITRO 7.0.0: outlev=3
maxit=100
opttol=1e-6
feastol=1e-6

Problem Characteristics
-----------------------
Objective goal:  Maximize
Number of variables:                     2408
    bounded below:                          6
    bounded above:                          0
    bounded below and above:             1600
    fixed:                                  0
    free:                                 802
Number of constraints:                   2400
    linear equalities:                    800
    nonlinear equalities:                1600
    linear inequalities:                    0
    nonlinear inequalities:                 0
    range:                                  0
Number of nonzeros in Jacobian:        155520
Number of nonzeros in Hessian:         672031
```

AMPL/KNITRO Output

```
  Iter      Objective      FeasError   OptError    ||Step||    CGits
--------  --------------  ----------  ----------  ----------  -------
       0   -6.644466e+03   7.713e-01
       1   -6.662212e+03   4.133e-01   9.244e+01   1.161e+02        0
       2   -6.635552e+03   5.440e-02   1.462e+02   5.754e+01        0
       3   -6.630317e+03   4.184e-03   1.983e+01   1.585e+01        0
       4   -6.630208e+03   8.762e-05   2.569e-01   3.576e+00        0
       5   -6.630209e+03   5.055e-08   6.440e-04   3.787e-02        0
       6   -6.630209e+03   1.260e-11   9.450e-09   2.260e-05        0

EXIT: Locally optimal solution found.
```

AMPL/KNITRO Output

```
Final Statistics
----------------
Final objective value               =  -6.63020945176702e+03
Final feasibility error (abs / rel) =   1.26e-11 / 1.26e-11
Final optimality error  (abs / rel) =   9.45e-09 / 7.94e-11
# of iterations                     =          6
# of CG iterations                  =          0
# of function evaluations           =          7
# of gradient evaluations           =          7
# of Hessian evaluations            =          6
Total program time (secs)           =       19.67583 (    19.612 CPU time)
Time spent in evaluations (secs)    =       15.54939

===============================================================================

KNITRO 7.0.0: Locally optimal solution.
objective -6630.209452; feasibility error 1.26e-11
6 iterations; 7 function evaluations
```

Solving All Equilibria in ML Estimation?

- It has been stated in the literature that in ML estimation, researchers must solve for all the Markov perfect equilibria at each guess of structural parameter vector.
- This statement is true only when the nested-fixed point algorithm is used to compute the ML estimator
- When using the constrained optimization approach, researchers do not need to solve for all the equilibria at each guess of structural parameter vector
    - constraints are satisfied (and an equilibrium solved) only at a solution, not at every iteration
    - 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
    - These two features eliminate a large set of equilibria together with structural parameters that do not need to be solved

Two-Step Methods: Intuition

- Recall the constrained optimization formulation for the ML estimator is

$$\begin{array}{ll} \max\limits_{(\boldsymbol{\theta},\boldsymbol{P},\boldsymbol{V})} & \mathcal{L}\left(\boldsymbol{Z},\boldsymbol{P}\right) \\ \text{subject to} & \boldsymbol{V} = \Psi^{\boldsymbol{V}}\left(\boldsymbol{V},\boldsymbol{P},\boldsymbol{\theta}\right) \\ & \boldsymbol{P} = \Psi^{\boldsymbol{P}}\left(\boldsymbol{V},\boldsymbol{P},\boldsymbol{\theta}\right) \end{array}$$

- Denote the solution by $(\boldsymbol{\theta}^*, \boldsymbol{P}^*, \boldsymbol{V}^*)$
- Suppose we know $\boldsymbol{P}^*$, how do we recover $\boldsymbol{\theta}^*$ (and $\boldsymbol{V}^*$)?

Two-Step Pseudo Maximum-Likelihood (2S-PML)

- Step 1: nonparametrically estimate the conditional choice probabilities, denoted by $\widehat{\boldsymbol{P}}$ directly from the observed data $\boldsymbol{Z}$
- Step 2: Solve

$$\begin{array}{cl} \max\limits_{(\boldsymbol{\theta},\boldsymbol{P},\boldsymbol{V})} & \mathcal{L}\left(\boldsymbol{Z},\boldsymbol{P}\right) \\ \text{subject to} & \boldsymbol{V} = \Psi^{\boldsymbol{V}}\left(\boldsymbol{V},\widehat{\boldsymbol{P}},\boldsymbol{\theta}\right) \\ & \boldsymbol{P} = \Psi^{\boldsymbol{P}}\left(\boldsymbol{V},\widehat{\boldsymbol{P}},\boldsymbol{\theta}\right) \end{array}$$

or, equivalently,

$$\begin{array}{cl} \max\limits_{(\boldsymbol{\theta},\boldsymbol{V})} & \mathcal{L}\left(\boldsymbol{Z},\Psi^{\boldsymbol{P}}\left(\boldsymbol{V},\widehat{\boldsymbol{P}},\boldsymbol{\theta}\right)\right) \\ \text{subject to} & \boldsymbol{V} = \Psi^{\boldsymbol{V}}\left(\boldsymbol{V},\widehat{\boldsymbol{P}},\boldsymbol{\theta}\right) \end{array}$$

Reformulation of the Optimization Problem in Step 2

- **Bellman Optimality**. $\forall i \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}$

$$V_i(\boldsymbol{x}) = \sum_{a_i \in \mathcal{A}} P_i(a_i|\boldsymbol{x}) \left[\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta}) + e_i^{\boldsymbol{P}}(a_i, \boldsymbol{x})\right] + \beta \sum_{\boldsymbol{x}' \in \mathcal{X}} V_i(\boldsymbol{x}') f_{\mathcal{X}}^{\boldsymbol{P}}(\boldsymbol{x}'|\boldsymbol{x})$$

Reformulation of the Optimization Problem in Step 2

- **Bellman Optimality**. $\forall i \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}$

$$V_i(\boldsymbol{x}) = \sum_{a_i \in \mathcal{A}} P_i(a_i|\boldsymbol{x}) \left[\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta}) + e_i^{\boldsymbol{P}}(a_i, \boldsymbol{x})\right] + \beta \sum_{\boldsymbol{x}' \in \mathcal{X}} V_i(\boldsymbol{x}') f_{\mathcal{X}}^{\boldsymbol{P}}(\boldsymbol{x}'|\boldsymbol{x})$$

- Define $\boldsymbol{V}_i = [V_i(x)]_{\boldsymbol{x} \in \mathcal{X}}$, $\widehat{\boldsymbol{P}}_i(a_i) = [\widehat{P}_i(a_i|\boldsymbol{x})]_{\boldsymbol{x}}$, $\boldsymbol{e}_i^{\widehat{\boldsymbol{P}}}(a_i) = [e_i^{\widehat{\boldsymbol{P}}}(a_i, \boldsymbol{x})]_{\boldsymbol{x}}$, $\boldsymbol{\pi}_i(a_i, \boldsymbol{\theta}) = [\pi_i(a_i|\boldsymbol{x}, \boldsymbol{\theta})]_{\boldsymbol{x}}$, and $\boldsymbol{F}_{\mathcal{X}}^{\widehat{\boldsymbol{P}}} = \left[f_{\mathcal{X}}^{\widehat{\boldsymbol{P}}}(\boldsymbol{x}'|\boldsymbol{x})\right]_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}}$

Reformulation of the Optimization Problem in Step 2

- **Bellman Optimality**. $\forall i \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}$

$$V_i\left(\boldsymbol{x}\right) = \sum_{a_i \in \mathcal{A}} P_i\left(a_i|\boldsymbol{x}\right)\left[\pi_i\left(a_i|\boldsymbol{x},\boldsymbol{\theta}\right) + e_i^{\boldsymbol{P}}\left(a_i,\boldsymbol{x}\right)\right] + \beta \sum_{\boldsymbol{x}' \in \mathcal{X}} V_i\left(\boldsymbol{x}'\right) f_{\mathcal{X}}^{\boldsymbol{P}}\left(\boldsymbol{x}'|\boldsymbol{x}\right)$$

- Define $\boldsymbol{V}_i = [V_i(x)]_{\boldsymbol{x}\in\mathcal{X}}$, $\widehat{\boldsymbol{P}}_i(a_i) = [\widehat{P}_i(a_i|\boldsymbol{x})]_{\boldsymbol{x}}$, $\boldsymbol{e}_i^{\widehat{\boldsymbol{P}}}(a_i) = [e_i^{\widehat{\boldsymbol{P}}}(a_i,\boldsymbol{x})]_{\boldsymbol{x}}$, $\boldsymbol{\pi}_i(a_i,\boldsymbol{\theta}) = [\pi_i(a_i|\boldsymbol{x},\boldsymbol{\theta})]_{\boldsymbol{x}}$, and $\boldsymbol{F}_{\mathcal{X}}^{\widehat{\boldsymbol{P}}} = \left[f_{\mathcal{X}}^{\widehat{\boldsymbol{P}}}\left(\boldsymbol{x}'|\boldsymbol{x}\right)\right]_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}}$
- The Bellman equation above can be rewritten as

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

or equivalently

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

or in a compact notation

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

Reformulation of the Optimization Problem in Step 2

- Replacing the constraint $\boldsymbol{V} = \Psi^{\boldsymbol{V}}\left(\boldsymbol{V}, \widehat{\boldsymbol{P}}, \boldsymbol{\theta}\right)$ by $\boldsymbol{V} = \boldsymbol{\Gamma}(\boldsymbol{\theta}, \widehat{\boldsymbol{P}})$ through a simple elimination of variables $\boldsymbol{V}$, the optimization problem in Step 2 becomes

$$\max_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{Z}, \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\boldsymbol{\theta}, \widehat{\boldsymbol{P}}), \widehat{\boldsymbol{P}}, \boldsymbol{\theta}\right)\right).$$

- The 2S-PML estimator is defined as

$$\boldsymbol{\theta}^{2S-PML} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{Z}, \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\boldsymbol{\theta}, \widehat{\boldsymbol{P}}), \widehat{\boldsymbol{P}}, \boldsymbol{\theta}\right)\right).$$

NPL Estimator

- The 2S-PML estimator can have large biases in finite samples
- In an effort to reduce the finite-sample biases associated with the 2S-PML estimator, Aguirregabiria and Mira (2007) propose an NPL estimator
- An NPL fixed point $(\tilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{P}})$ satisfies the conditions:

$$\tilde{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathcal{L}\left(\boldsymbol{Z}, \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\boldsymbol{\theta}, \widetilde{\boldsymbol{P}}), \widetilde{\boldsymbol{P}}, \boldsymbol{\theta}\right)\right)$$
$$\widetilde{\boldsymbol{P}} = \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{P}}), \widetilde{\boldsymbol{P}}, \tilde{\boldsymbol{\theta}}\right)$$

- The NPL algorithm: For $1 \le K \le \bar{K}$, iterate over Steps 1 and 2

Step 1. Given $\widetilde{\boldsymbol{P}}_{K-1}$,
solve $\tilde{\boldsymbol{\theta}}_K = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathcal{L}\left(\boldsymbol{Z}, \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\boldsymbol{\theta}, \widetilde{\boldsymbol{P}}_{K-1}), \widetilde{\boldsymbol{P}}_{K-1}, \boldsymbol{\theta}\right)\right)$.

Step 2. Given $\tilde{\boldsymbol{\theta}}_K$, update $\widetilde{\boldsymbol{P}}_K$ by
$\widetilde{\boldsymbol{P}}_K = \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\boldsymbol{P}}_{K-1}), \widetilde{\boldsymbol{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\right)$; increase K by 1.

A Modified NPL Algorithm: NPL-Λ

- It is now well known that the NPL algorithm may not converge or even if it converges, it may fail to provide consistent estimates
- Kasahara and Shimotsu (2012) propose the NPL-Λ algorithm that modifies Step 2 of the NPL algorithm to compute the NPL estimator

$$\widetilde{\boldsymbol{P}}_K = \left(\Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\boldsymbol{P}}_{K-1}), \widetilde{\boldsymbol{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\right)\right)^{\lambda} \left(\widetilde{\boldsymbol{P}}_{K-1}\right)^{1-\lambda}$$

  where λ is chosen to be between 0 and 1.
- The proper value for λ depends on the true parameter values $\boldsymbol{\theta}^0$
- Alternatively, Kasahara and Shimotsu suggest computing the spectral radius (largest eigenvalue) of the mapping $\nabla_{\boldsymbol{P}} \Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\tilde{\boldsymbol{\theta}}_K, \widetilde{\boldsymbol{P}}_{K-1}), \widetilde{\boldsymbol{P}}_{K-1}, \tilde{\boldsymbol{\theta}}_K\right)$ at every guess of structural parameter vector $\tilde{\boldsymbol{\theta}}_K$

Experiment Design

- Three experiment specifications with two cases in each experiment
- Experiment 1: Kasahara and Shimotsu (2012) example
- Experiment 2: Aguirregabiria and Mira (2007) example
- Experiment 3: Examples with increasing $|\mathcal{S}|$, the number of market size values
- Market size transition matrix is

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

Experiment 2: Aguirregabiria and Mira (2007) Example

- $N = 5$ players
- $\mathcal{S} = \{1, 2, \ldots, 5\}$
- Total number of grid points in the state space: $|\mathcal{X}| = |\mathcal{S}| \times |\mathcal{A}|^N = 5 \times 2^5 = 160$
- The discount factor $\beta = 0.95$; the scale parameter of the type-I extreme value distribution $\sigma = 1$
- The common-knowledge component of the per-period payoff

$$\Pi_i\left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t; \boldsymbol{\theta}\right)$$

$$= \begin{cases} \theta_{RS} s^t - \theta_{RN} \log\left(1 + \sum_{j \neq i} a_j^t\right) - \theta_{FC,i} - \theta_{EC}\left(1 - a_i^{t-1}\right), & \text{if } a_i^t = 1, \\ 0 & \text{if } a_i^t = 0, \end{cases}$$

- $\boldsymbol{\theta} = (\theta_{RS}, \theta_{RN}, \boldsymbol{\theta}_{FC}, \theta_{EC})$: the vector of structural parameters with $\boldsymbol{\theta}_{FC} = \{\theta_{FC,i}\}_{i=1}^N$

Experiment 2: Cases 3 and 4

- True values of structural parameters $\boldsymbol{\theta}^0_{FC} = (1.9, 1.8, 1.7, 1.6, 1.5)$ and $\theta^0_{EC} = 1$
- Consider two sets of true parameter values for θ_{RS} and θ_{RN}

$$\begin{aligned} \text{Case 3:} \quad & (\theta^0_{RN}, \theta^0_{RS}) = (2, 1); \\ \text{Case 4:} \quad & (\theta^0_{RN}, \theta^0_{RS}) = (4, 2). \end{aligned}$$

- Case 3 is Experiment 3 in Aguirregabiria and Mira (2007)
- The ML estimator solves the constrained optimization problem with 2,400 constraints and 2,408 variables.

Experiment 3: Cases 5 and 6

- Consider two sets of market size values:

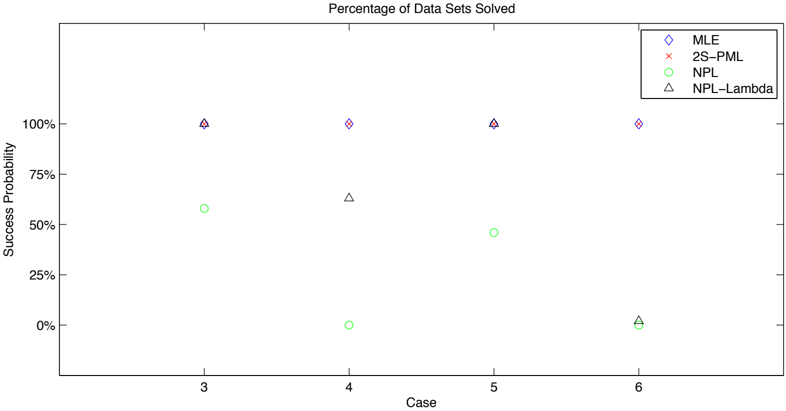
$$\text{Case 5:} \quad |\mathcal{S}| = 10 \text{ with } \mathcal{S} = \{1, 2, \ldots, 10\};$$
$$\text{Case 6:} \quad |\mathcal{S}| = 15 \text{ with } \mathcal{S} = \{1, 2, \ldots, 15\}.$$

- All other specifications remain the same as those in Case 3 in Experiment 2
- Case 5: The ML estimator solves the constrained optimization problem with 4,800 constraints and 4,808 variables.
- Case 6: The ML estimator solves the constrained optimization problem with 7,200 constraints and 7,208 variables.

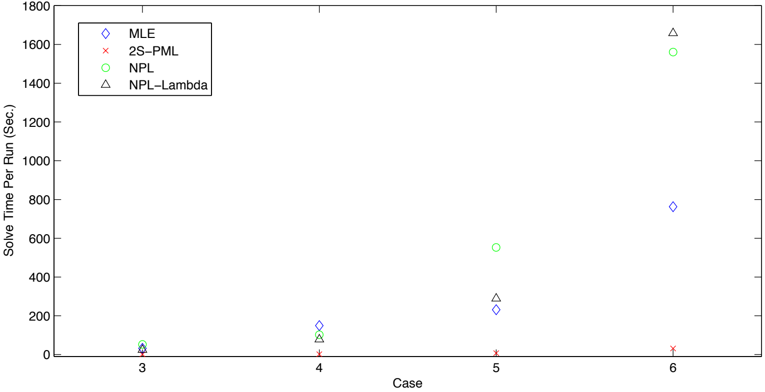
Data Simulation and Algorithm Implementation

- In each data set: $M = 400$ and $T = 10$
- For Case 3 and 4 in Experiments 2
    - Construct 100 data sets for each case
    - 10 starting points for each data set
- For Cases 5 and 6 in Experiments 3
    - Construct 50 data sets for each case
    - 5 start points for each data sets
- For NPL and NPL-Λ: $\bar{K} = 100$
- For the NPL-Λ algorithm: $\lambda = 0.5$

Monte Carlo Results: Percentage of Data Sets Solved



Monte Carlo Results: Avg. Solve Time Per Run



Monte Carlo Results: Estimates for Experiment 2

Case	Estimator	Estimates							
		$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
	Truth	1.9	1.8	1.7	1.6	1.5	1	2	1
3	MLE	1.895 (0.077)	1.794 (0.078)	1.697 (0.075)	1.597 (0.074)	1.495 (0.073)	0.990 (0.046)	2.048 (0.345)	1.011 (0.095)
3	2S-PML	1.884 (0.066)	1.774 (0.069)	1.662 (0.065)	1.548 (0.062)	1.425 (0.057)	1.040 (0.039)	0.805 (0.251)	0.671 (0.068)
3	NPL	1.894 (0.075)	1.788 (0.077)	1.688 (0.069)	1.581 (0.071)	1.478 (0.073)	1.010 (0.041)	1.812 (0.213)	0.946 (0.061)
3	NPL-Λ	1.896 (0.077)	1.795 (0.079)	1.697 (0.076)	1.597 (0.074)	1.495 (0.073)	0.991 (0.044)	2.039 (0.330)	1.008 (0.091)
	Truth	1.9	1.8	1.7	1.6	1.5	1	4	2
4	MLE	1.897 (0.084)	1.797 (0.084)	1.697 (0.082)	1.594 (0.085)	1.496 (0.095)	0.993 (0.045)	4.015 (0.216)	2.004 (0.086)
4	2S-PML	1.934 (0.090)	1.824 (0.085)	1.703 (0.079)	1.556 (0.079)	1.338 (0.085)	1.123 (0.049)	2.297 (0.330)	1.409 (0.117)
4	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)
4	NPL-Λ	1.900 (0.079)	1.801 (0.081)	1.700 (0.077)	1.600 (0.080)	1.500 (0.091)	0.991 (0.052)	4.023 (0.255)	2.007 (0.098)

Monte Carlo Results: Estimates for Experiment 3

$\|S\|$	Estimator	Estimates							
		$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
	Truth	1.9	1.8	1.7	1.6	1.5	1	2	1
10	MLE	1.882 (0.092)	1.780 (0.087)	1.677 (0.079)	1.584 (0.084)	1.472 (0.068)	0.999 (0.046)	2.031 (0.201)	1.004 (0.048)
10	2S-PML	1.884 (0.102)	1.792 (0.088)	1.679 (0.082)	1.583 (0.087)	1.469 (0.068)	1.039 (0.048)	1.065 (0.222)	0.755 (0.053)
10	NPL	1.919 (0.092)	1.810 (0.089)	1.699 (0.068)	1.606 (0.079)	1.485 (0.071)	1.011 (0.050)	1.851 (0.136)	1.966 (0.036)
10	NPL-Λ	1.884 (0.095)	1.781 (0.089)	1.678 (0.081)	1.584 (0.085)	1.472 (0.070)	0.997 (0.049)	2.032 (0.211)	1.005 (0.051)
15	MLE	1.897 (0.098)	1.800 (0.107)	1.694 (0.087)	1.597 (0.093)	1.492 (0.090)	0.983 (0.059)	2.040 (0.311)	1.011 (0.069)
15	2S-PML	1.792 (0.119)	1.705 (0.123)	1.595 (0.119)	1.506 (0.114)	1.394 (0.114)	1.046 (0.059)	0.766 (0.220)	0.664 (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 (0.000)	1.821 (0.000)	1.671 (0.000)	1.611 (0.000)	1.531 (0.000)	1.012 (0.000)	1.992 (0.000)	1.007 (0.000)

Final Comment: Lyapunov-Stable Equilibria?

- Aguirregabiria and Nevo (2012) have argued that with multiple equilibria, it is reasonable to assume that only Lyapunov-stable (or best-response stable) equilibria will be played in the data, in which case the NPL algorithm should converge
- Lyapunov-stable (or best-response stable) equilibria:

$$\rho\left(\nabla_{\boldsymbol{P}}\Psi^{\boldsymbol{P}}\left(\boldsymbol{\Gamma}(\boldsymbol{\theta}^0,\boldsymbol{P}^0),\boldsymbol{P}^0,\boldsymbol{\theta}^0\right)\right) < 1$$

- The spectral radius of the mapping above depends not only on $\boldsymbol{\theta}^0$ but also on the grid of the market size values, market size transition, etc

Conclusion

- Recursive methods (NPL and NPL-Λ algorithms) are not reliable computational algorithms and should be used with caution
- The 2S-PML estimator often produces large finite-sample biases
  - Not surprising, given the comment in Pakes, Ostrovsky, and Berry (2007)
  - Can other two-step estimators perform better?
- The constrained optimization approach is reliable and capable of solving empirically relevant dynamic game models such as those in Aguirregabiria and Mira (2007)
- Improving the performance of the constrained optimization approach on dynamic games with higher-dimensional state space?

Detailed Numerical Results

Data Simulation and Algorithm Implementation

- For each case, we find only one equilibrium at the true parameter values (using KNITRO with 100 starting points)
- For each case in Experiments 1 and 2
  - Data sets with three sizes: $M = 400$; $T = 1, 10$, and 20
  - Construct 100 data sets for each case
  - 10 starting points for each data set
- For Cases 5 and 6 in Experiments 3
  - Data sets with $M = 400$ and $T = 10$
  - Construct 50 data sets for each case
  - 5 start points for each data sets
- For NPL and NPL-Λ
  - Experiment 1: $\bar{K} = 250$
  - Experiments 2 and 3: $\bar{K} = 100$
- For the NPL-Λ algorithm: $\lambda = 0.5$

Experiment 1: Kasahara and Shimotsu (2012) Example

- $N = 3$ players
- $\mathcal{S} = \{2, 6, 10\}$
- Total number of grid points in the state space: $|\mathcal{X}| = |\mathcal{S}| \times |\mathcal{A}|^N = 3 \times 2^3 = 24$
- The common-knowledge component of the per-period payoff

$$
\Pi_i \left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t; \boldsymbol{\theta} \right)
$$

$$
= \begin{cases} \theta^{RS} \log\left(s^t\right) - \theta^{RN} \log\left(1 + \sum_{j \neq i} a_j^t\right) - \theta_i^{FC} - \theta^{EC}\left(1 - a_i^{t-1}\right), & \text{if } a_i^t = 1, \\ 0 & \text{if } a_i^t = 0, \end{cases}
$$

- $\boldsymbol{\theta} = \left(\theta^{RS}, \theta^{RN}, \boldsymbol{\theta}^{FC}, \theta^{EC}\right)$: the vector of structural parameters with $\boldsymbol{\theta}^{FC} = \{\theta_i^{FC}\}_{i=1}^N$

Experiment 1: Cases 1 and 2

- The discount factor $\beta = 0.96$; the scale parameter of the type-I extreme value distribution $\sigma = 1$
- Values of structural parameters $\boldsymbol{\theta}^{FC} = (1.0, 0.9, 0.8)$ and $\theta^{EC} = 1$ are fixed; estimate only θ^{RS} and θ^{RN}
- Consider two sets of parameter values for θ^{RS} and θ^{RN}

$$\begin{aligned} \text{Case 1:} \quad & (\theta^{RN}, \theta^{RS}) = (2, 1); \\ \text{Case 2:} \quad & (\theta^{RN}, \theta^{RS}) = (4, 1). \end{aligned}$$

- The ML estimator solves the constrained optimization problem with 216 constraints and 218 variables.

Monte Carlo Results for Experiment 1: Case 1

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

Monte Carlo Results for Experiment 1: Case 2

M	T	Estimator	Estimates		CPU Time (in sec.)	Data Sets Converged	Runs Converged	Avg. NPL(-Λ) Iter.
			θ_{RN}	θ_{RS}				
		Truth	4	1	–	–	–	–
400	1	MLE	4.055 (0.613)	1.003 (0.158)	0.61	100	735	–
400	1	2S-PML	3.107 (0.442)	0.839 (0.099)	0.02	100	1000	–
400	1	NPL	N/A (N/A)	N/A (N/A)	1.68	0	0	250
400	1	NPL-Λ	N/A (N/A)	N/A (N/A)	1.68	0	0	250
400	10	MLE	4.003 (0.039)	1.000 (0.016)	0.50	100	767	–
400	10	2S-PML	3.902 (0.099)	0.983 (0.025)	0.04	100	1000	–
400	10	NPL	N/A (N/A)	N/A (N/A)	7.61	0	0	250
400	10	NPL-Λ	N/A (N/A)	N/A (N/A)	7.54	0	0	250
400	20	MLE	4.003 (0.032)	1.001 (0.011)	0.47	100	820	–
400	20	2S-PML	3.954 (0.084)	0.992 (0.019)	0.06	100	1000	–
400	20	NPL	N/A (N/A)	N/A (N/A)	12.38	0	0	250
400	20	NPL-Λ	N/A (N/A)	N/A (N/A)	12.41	0	0	250

Summary of Monte Carlo Results: Experiment 1

- Case 1: $(\theta_{RN}, \theta_{RS}) = (2, 1)$
  - All estimation algorithms converged for all data sets
  - All estimators produced fairly precise estimates, except for the 2S-PML estimator with $T = 1$
  - These estimates become more precise as T increases
- Case 2: $(\theta_{RN}, \theta_{RS}) = (4, 1)$
  - Both NPL and NPL-Λ failed to converge for all data sets ($\bar{K} = 250$)
  - Both 2S-PML and the constrained optimization approach converged for all 100 data sets, but the constrained optimization approach converged for only 735 (out of 1000) runs for $T = 1$, and 820 runs for $T = 20$
  - For $T = 1$, 2S-PML estimates $(\theta_{RN}, \theta_{RS}) = (3.107, 0.839)$

Experiment 2: Aguirregabiria and Mira (2007) Example

- $N = 5$ players
- $\mathcal{S} = \{1, 2, \ldots, 5\}$
- Total number of grid points in the state space: $|\mathcal{X}| = |\mathcal{S}| \times |\mathcal{A}|^N = 5 \times 2^5 = 160$
- The common-knowledge component of the per-period payoff

$$
\begin{aligned}
&\Pi_i\left(a_i^t, \boldsymbol{a}_{-i}^t, \boldsymbol{x}^t; \boldsymbol{\theta}\right) \\
= &\begin{cases} \theta^{RS} \boldsymbol{s}^t - \theta^{RN} \log\left(1 + \sum\limits_{j \neq i} a_j^t\right) - \theta_i^{FC} - \theta^{EC}\left(1 - a_i^{t-1}\right), & \text{if } a_i^t = 1, \\ 0 & \text{if } a_i^t = 0, \end{cases}
\end{aligned}
$$

- $\boldsymbol{\theta} = \left(\theta^{RS}, \theta^{RN}, \boldsymbol{\theta}^{FC}, \theta^{EC}\right)$: the vector of structural parameters with $\boldsymbol{\theta}^{FC} = \{\theta_i^{FC}\}_{i=1}^N$

Experiment 2: Cases 3 and 4

- The discount factor $\beta = 0.95$; the scale parameter of the type-I extreme value distribution $\sigma = 1$
- True values of structural parameters $\boldsymbol{\theta}^{FC} = (1.9, 1.8, 1.7, 1.6, 1.5)$ and $\theta^{EC} = 1$
- Consider two sets of true parameter values for θ^{RS} and θ^{RN}

$$\begin{aligned} &\text{Case 3:} \quad (\theta^{RN}, \theta^{RS}) = (2, 1); \\ &\text{Case 4:} \quad (\theta^{RN}, \theta^{RS}) = (4, 2). \end{aligned}$$

- Case 3 is Experiment 3 in Aguirregabiria and Mira (2007)
- The ML estimator solves the constrained optimization problem with 2,400 constraints and 2,408 variables.

Monte Carlo Results for Experiment 2: Case 3

M	T	Estimator	CPU Time (in sec.)	Data Sets Converged	Runs Converged	Avg. NPL(-Λ) Iter.
400	1	MLE	216.31	100	736	–
400	1	2S-PML	1.34	100	1000	–
400	1	NPL	45.85	53	530	64.44
400	1	NPL-Λ	36.78	90	882	49.39
400	10	MLE	32.11	100	995	–
400	10	2S-PML	1.40	100	1000	–
400	10	NPL	52.39	58	580	72.14
400	10	NPL-Λ	24.31	100	1000	33.46
400	20	MLE	29.74	100	999	–
400	20	2S-PML	1.54	100	1000	–
400	20	NPL	55.27	67	664	71.54
400	20	NPL-Λ	23.75	100	1000	31.50

Monte Carlo Results for Experiment 2: Case 3

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

Monte Carlo Results for Experiment 2: Case 4

M	T	Estimator	CPU Time (in sec.)	Data Sets Converged	Runs Converged	Avg. NPL(-Λ) Iter.
400	1	MLE	273.88	100	582	–
400	1	2S-PML	1.71	100	1000	–
400	1	NPL	103.77	2	20	99.47
400	1	NPL-Λ	74.22	84	840	69.67
400	10	MLE	149.65	100	812	–
400	10	2S-PML	1.59	100	1000	–
400	10	NPL	102.69	0	0	100
400	10	NPL-Λ	78.12	63	630	77.28
400	20	MLE	121.71	100	871	–
400	20	2S-PML	1.67	100	1000	–
400	20	NPL	107.07	0	0	100
400	20	NPL-Λ	84.30	53	530	79.30

Monte Carlo Results for Experiment 2: Case 4

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

Summary of Monte Carlo Results: Experiment 2

- Case 3: $(\theta_{RN}, \theta_{RS}) = (2, 1)$
  - This is Experiment 3 in Aguirregabiria and Mira (2007)
  - The NPL-Λ algorithm worked quite well, converging for 90 data sets for $T = 1$ and all 100 data sets for $T = 10$ and 20
  - The mean estimates of the 2S-PML estimator for parameters θ_{RN} and θ_{RS} are quite biased
  - The constrained optimization approach converged for all 100 data sets for each T. However, it was slow for $T = 1$, needing 216 seconds per run;
  - For $T = 1$, mean estimates of the constrained optimization approach are more biased than those of the NPL-Λ algorithm. However, it yielded higher likelihood values than the NPL-Λ algorithm for all 100 data sets

Summary of Monte Carlo Results: Experiment 2

- Case 4: $(\theta_{RN}, \theta_{RS}) = (4, 2)$
  - NPL converged for 2 data set for $T = 1$ and failed for all 100 data sets for $T = 10$ and 20
  - The NPL-Λ algorithm performed better than the NPL algorithm, but failed more frequently than it did in Case 3, converging in 84 data sets for $T = 1$ and only 53 data sets for $T = 20$
  - The 2S-PML estimator produced inaccurate estimates of parameters θ_{RN} and θ_{RS}
  - The constrained optimization approach converged for all 100 data sets for different T, although it converged for only 582 runs (out of 1,000) for $T = 1$; it also produced fairly accurate estimates of all structural parameters

Experiment 3: Cases 5 and 6

- Consider two sets of market size values:

$$\text{Case 5:} \quad |\mathcal{S}| = 10 \text{ with } \mathcal{S} = \{1, 2, \dots, 10\};$$
$$\text{Case 6:} \quad |\mathcal{S}| = 15 \text{ with } \mathcal{S} = \{1, 2, \dots, 15\}.$$

- All other specifications remain the same as those in Case 3 in Experiment 2
- Case 5: The ML estimator solves the constrained optimization problem with 4,800 constraints and 4,808 variables.
- Case 6: The ML estimator solves the constrained optimization problem with 7,200 constraints and 7,208 variables.

Monte Carlo Results for Experiment 3: Cases 5 and 6

$\|S\|$	Estimator	CPU Time (in sec.)	Data Sets Converged	Runs Converged	Avg. NPL(-Λ) Iter.
10	MLE	231.52	50	240	–
10	2S-PML	7.44	50	250	–
10	NPL	552.85	23	76	89
10	NPL-Λ	289.39	50	241	43
15	MLE	762.78	50	222	–
15	2S-PML	31.45	50	242	–
15	NPL	1560.08	0	0	100
15	NPL-Λ	1658.08	1	3	99.7

Monte Carlo Results for Experiment 3: Cases 5 and 6

$\lvert S \rvert$	Estimator	Estimates							
		$\theta_{FC,1}$	$\theta_{FC,2}$	$\theta_{FC,3}$	$\theta_{FC,4}$	$\theta_{FC,5}$	θ_{EC}	θ_{RN}	θ_{RS}
	Truth	1.9	1.8	1.7	1.6	1.5	1	2	1
10	MLE	1.882 (0.092)	1.780 (0.087)	1.677 (0.079)	1.584 (0.084)	1.472 (0.068)	0.999 (0.046)	2.031 (0.201)	1.004 (0.048)
10	2S-PML	1.884 (0.102)	1.792 (0.088)	1.679 (0.082)	1.583 (0.087)	1.469 (0.068)	1.039 (0.048)	1.065 (0.222)	0.755 (0.053)
10	NPL	1.919 (0.092)	1.810 (0.089)	1.699 (0.068)	1.606 (0.079)	1.485 (0.071)	1.011 (0.050)	1.851 (0.136)	1.966 (0.036)
10	NPL-Λ	1.884 (0.095)	1.781 (0.089)	1.678 (0.081)	1.584 (0.085)	1.472 (0.070)	0.997 (0.049)	2.032 (0.211)	1.005 (0.051)
15	MLE	1.897 (0.098)	1.800 (0.107)	1.694 (0.087)	1.597 (0.093)	1.492 (0.090)	0.983 (0.059)	2.040 (0.311)	1.011 (0.069)
15	2S-PML	1.792 (0.119)	1.705 (0.123)	1.595 (0.119)	1.506 (0.114)	1.394 (0.114)	1.046 (0.059)	0.766 (0.220)	0.664 (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 (0.000)	1.821 (0.000)	1.671 (0.000)	1.611 (0.000)	1.531 (0.000)	1.012 (0.000)	1.992 (0.000)	1.007 (0.000)

Summary of Monte Carlo Results: Experiment 3

- Case 5: $|\mathcal{S}| = 10$
  - The NPL algorithm converged for only 23 of 50 data sets (or 76 out of 250 runs), and produced highly biased estimates of the parameter θ_{RS} with a mean estimate of 1.966 (true value is 1)
  - The NPL-Λ algorithm converged for all 50 data sets
- Case 6: $|\mathcal{S}| = 15$
  - The NPL algorithm failed to converge for all 50 data sets
  - The NPL-Λ algorithm converged for only 1 of 50 data sets
  - The 2S-PML estimator produced highly biased estimates
  - The constrained optimization approach converged for all 50 data sets and produced accurate estimates