Constrained Optimization Approaches to Estimation of Structural Models

Che-Lin Su
The University of Chicago
Booth School of Business
Che-Lin.Su@ChicagoBooth.edu
This is joint work with Kenneth L. Judd

November 24, 2010

Outline

1. Estimation of Dynamic Programming Models
   - New Monte Carlo Results
2. Estimation of Games with Multiple Equilibria
   - Example of discrete-choice games

Structural Estimation Overview

- Great interest in estimating models based on economic structure
  - DP models of individual behavior: Rust (1987) – NFXP
  - Demand Estimation: BLP(1995), Nevo (2000), DFS (2010)
  - Nash equilibria of games – static, dynamic: BBL(2007), Ag-M (2007)
  - Auctions: Paarsch and Hong (2006), Hubbard and Paarsch (2008)
  - Dynamic stochastic general equilibrium
  - Popularity of structural models in empirical IO and marketing
  - Operations Management
- Model sophistication introduces computational difficulties
- General belief: Estimation is a major computational challenge because it involves solving the model many times

Current Views on Structural Estimation

Tulin Erdem, Kannan Srinivasan, Wilfred Amaldoss, Patrick Bajari, Hai Che, Teck Ho, Wes Hutchinson, Michael Katz, Michael Keane, Robert Meyer, and Peter Reiss, “Theory-Driven Choice Models”, *Marketing Letters* (2005)

> *Estimating structural models can be computationally difficult. For example, dynamic discrete choice models are commonly estimated using the nested fixed point algorithm (see Rust 1994). This requires solving a dynamic programming problem thousands of times during estimation and numerically minimizing a nonlinear likelihood function....[S]ome recent research ... proposes computationally simple estimators for structural models ... The estimators ... use a two-step approach. ....The two-step estimators can have drawbacks. First, there can be a loss of efficiency. .... Second, stronger assumptions about unobserved state variables may be required. .... However, two-step approaches are computationally light, often require minimal parametric assumptions and are likely to make structural models accessible to a larger set of researchers.*

Part I

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

- Rust (1987)
- Each bus comes in for repair once a month
  - Bus repairman sees mileage x_t at time t *since last engine overhaul*
  - Repairman chooses between overhaul and ordinary maintenance

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

  - Repairman solves DP:

$$V_\theta(x_t) = \sup_{\{f_t, f_{t+1}, \ldots\}} E \left\{ \sum_{j=t}^{\infty} \beta^{j-t} \left[u(x_j, f_j, \theta) + \varepsilon_j(f_j) \right] | x_t \right\}$$

- 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
  - Transition probabilities in mileages $p(x_{t+1} | x_t, d_t, \theta^p)$

Zurcher's Bus Engine Replacement Problem

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

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

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

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

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

Nested Fixed Point Algo: Rust (1987)

- Outer loop: Solve likelihood

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

  - 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

Concerns with NFXP – DFS (2009)

- Inner-loop error propagates into outer-loop function and derivatives
- NFXP needs to solve inner-loop exactly each stage of parameter search
  - 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 correct
  - $\epsilon_{in} = 1.e - 13$
  - finished the inner-loop with Newton's method

Stopping Rules – DFS (2009)

- Notations:
  - $\mathcal{L}(EV(\theta, \epsilon_{in}), \theta)$: the programmed outer loop objective function with ϵ_{in}
  - L: the Lipschitz constant of the inner-loop contraction mapping
- Analytic derivatives $\nabla_\theta \mathcal{L}(EV(\theta), \theta)$ is provided: $\epsilon_{out} = O(\frac{L}{1-L}\epsilon_{in})$
- Finite-difference derivatives are used: $\epsilon_{out} = O(\sqrt{\frac{L}{1-L}\epsilon_{in}})$

Constrained Optimization for Solving Zucher Model

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

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

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

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

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

- Solve constrained optimization problem

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

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

- Fixed point dimension: 175
- Maintenance cost function: $c(x, \theta_1) = 0.001 * \theta_{11} * x$
- Mileage transition: stay or move up at most 4 grid points
- True parameter values:
    - $\theta_{11} = 2.457$
    - $RC = 11.726$
    - $(\theta_{30}, \theta_{31}, \theta_{32}, \theta_{33}) = (0.0937, 0.4475, 0.4459, 0.0127)$
    - Solve for EV at the true parameter values
- Simulate 250 datasets of monthly data for 10 years and 50 buses
- Estimation implementations
    - 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: $\beta = 0.975$ and 0.980

β	Imple.	Parameters						MSE
		RC	θ_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	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_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	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_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	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

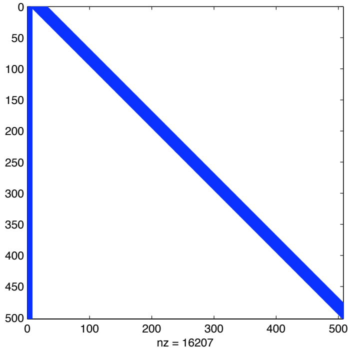
β	Imple.	Runs Conv.	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Contrac. Mapping Iter.
0.975	MPEC1	1240	0.13	12.8	17.6	–
	MPEC2	1247	7.9	53.0	62.0	–
	NFXP	998	24.6	55.9	189.4	$1.348e+5$
0.980	MPEC1	1236	0.15	14.5	21.8	–
	MPEC2	1241	8.1	57.4	70.6	–
	NFXP	1000	27.9	55.0	183.8	$1.625e+5$
0.985	MPEC1	1235	0.13	13.2	19.7	–
	MPEC2	1250	7.5	55.0	62.3	–
	NFXP	952	42.2	61.7	227.3	$2.658e+5$
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	$4.524e+5$
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	$7.485e+5$

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 more than 10000 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 II

Estimation of Games

Structural Estimation of Games

- An active research topic in Applied Econometrics/Empirical Industrial Organization
  - Agguiregabiria and Mira (2007), Bajari, Benkard, Levin (2007), Pesendorfer and Schmidt-Dengler (2008), Pakes, Ostrovsky, and Berry (2007), etc.
- 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: Prisoners Dilemma Game of Incomplete Information - due to John Rust

- Two players: a and b
- Actions: each player has two possible actions:

$$\begin{array}{ll} d_a = 1 & \text{if prisoner } a \text{ confess} \\ d_a = 0 & \text{if prisoner } a \text{ does not confess} \end{array}$$

Example: Prisoners Dilemma Game of Incomplete Information - due to John Rust

- Utility: Ex-post payoff to prisoners

$$u_a(d_a, d_b, x_a, \epsilon_a) = \theta^a_{d_a d_b} x_a + \sigma_a \epsilon_a(d_a)$$

$$u_b(d_a, d_b, x_b, \epsilon_b) = \theta^b_{d_a d_b} x_b + \sigma_b \epsilon_b(d_b)$$

  - $(\theta^a_{d_a d_b}, \theta^b_{d_a d_b})$ and (σ_a, σ_b): structural parameters to be estimated
  - (x_a, x_b): prisoners' observed types; **common knowledge**
  - (ϵ_a, ϵ_b): prisoners' unobserved types, **private information**
  - $(\epsilon_a(d_a), \epsilon_b(d_b))$ are observed only by each prisoner, but not by their opponent prisoner nor by the econometrician

Example: PD Game of Incomplete Information

- Assume the error terms (ϵ_a, ϵ_b) have a standardized type III extreme value distribution
- A Bayesian Nash equilibrium (p_a, p_b) satisfies

$$
\begin{aligned}
p_a &= \frac{1}{1 + exp\{x_a(\theta^a_{00} - \theta^a_{10})/\sigma_a + p_b x_a(\theta^a_{01} - \theta^a_{11} + \theta^a_{10} - \theta^a_{00})/\sigma_a\}} \\
&= \Psi_a(p_b, \theta^a, \sigma_a, x_a)
\end{aligned}
$$

$$
\begin{aligned}
p_b &= \frac{1}{1 + exp\{x_b(\theta^b_{00} - \theta^b_{01})/\sigma_b + p_a x_b(\theta^b_{10} - \theta^b_{11} + \theta^b_{01} - \theta^b_{00})/\sigma_b\}} \\
&= \Psi_b(p_a, \theta^b, \sigma_b, x_b)
\end{aligned}
$$

PD Example with One Market: Solving for Equilibria

- The true values of the structural parameters are

$$(\sigma_a, \sigma_b) = (0.1, 0.1)$$
$$(\theta^a_{11}, \theta^b_{11}) = (-2, -2) \qquad (\theta^a_{00}, \theta^b_{00}) = (-1, -1)$$
$$(\theta^a_{10}, \theta^b_{01}) = (-0.5, -0.5) \qquad (\theta^a_{01}, \theta^b_{10}) = (-0.9, -0.9)$$

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

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

PD 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

PD 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 $(\theta^a_{d_a d_b}, \theta^b_{d_a d_b})$ and (σ_a, σ_b)
- Identification: Can only identify four parameters

$$\begin{aligned} \alpha^a &= (\theta^a_{00} - \theta^a_{10})/\sigma_a, \qquad \alpha^b = (\theta^b_{00} - \theta^b_{01})/\sigma_b \\ \beta^a &= (\theta^a_{01} - \theta^a_{11})/\sigma_a, \qquad \beta^b = (\theta^b_{10} - \theta^b_{11})/\sigma_b \end{aligned}$$

- Impose symmetry condition for this example:

$$\alpha^a = \alpha^b = \alpha, \qquad \beta^a = \beta^b = \beta$$

PD Example: Maximum Likelihood Estimation

- Maximize the likelihood function

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

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

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

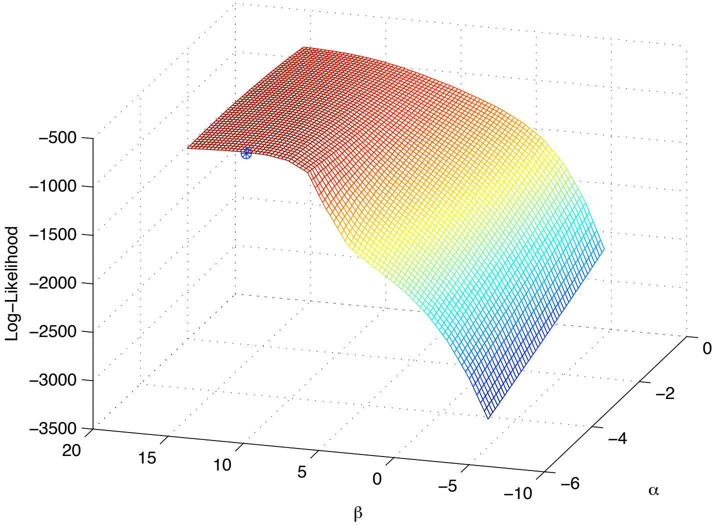
PD Example: MLE via NFXP

- 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:

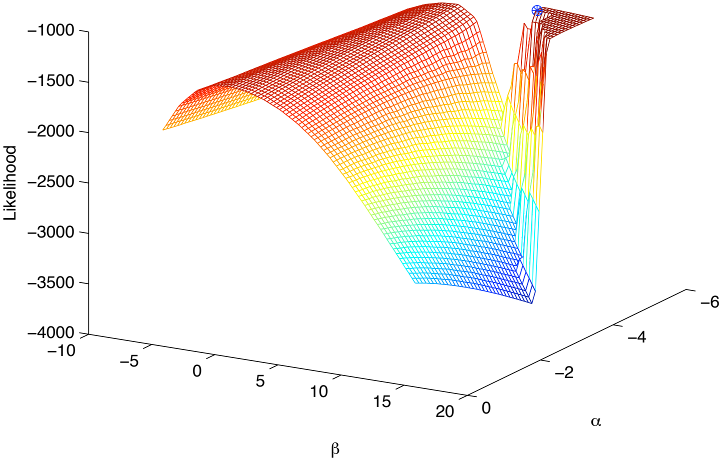
$$k^* = \underset{\{k=1,\ldots,K\}}{\operatorname{argmax}} \; log\mathcal{L}\left(p_a^k(\alpha, \beta), p_b^k(\alpha, \beta); X\right)$$

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

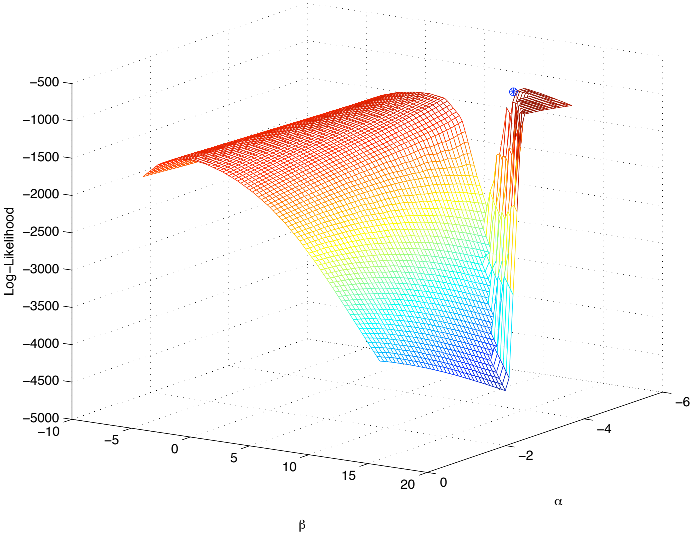
PD Example: Likelihood as a Function of (α, β) – Eq 1



PD Example: Likelihood as a Function of (α, β) – Eq 2



PD Example: Likelihood as a Function of (α, β) – Eq 3

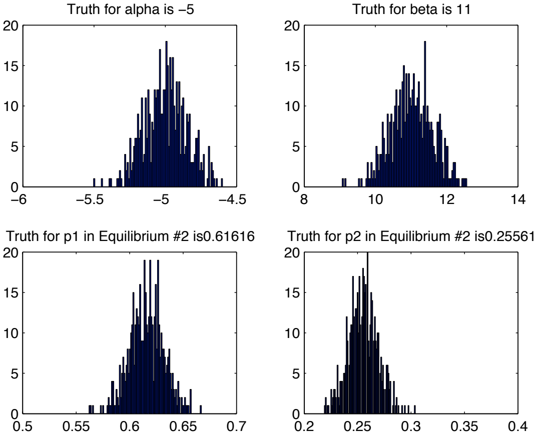


PD Example: Constrained Optimization Formulation for MLE Estimation

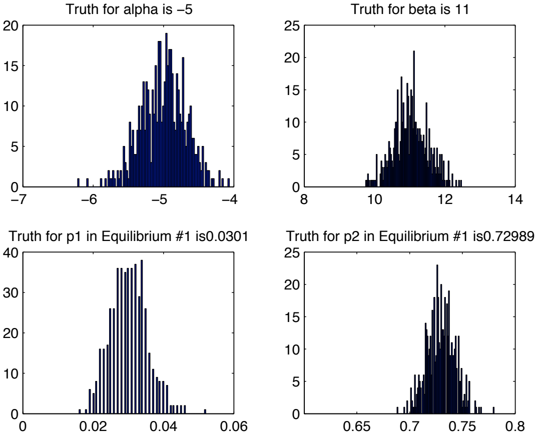
$$
\begin{aligned}
\max_{(\alpha,\beta,p_a,p_b)} \quad & log\mathcal{L}\left(p_a, p_b; X\right) \\
& = \sum_{i=1}^{1000} \left(d_a^i * \log(p_a) + (1 - d_a^i) * \log(1 - p_a) \right) \\
& + \sum_{i=1}^{1000} \left(d_b^i * \log(p_b) + (1 - d_b^i) * \log(1 - p_b) \right) \\
\text{subject to} \quad & p_a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

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

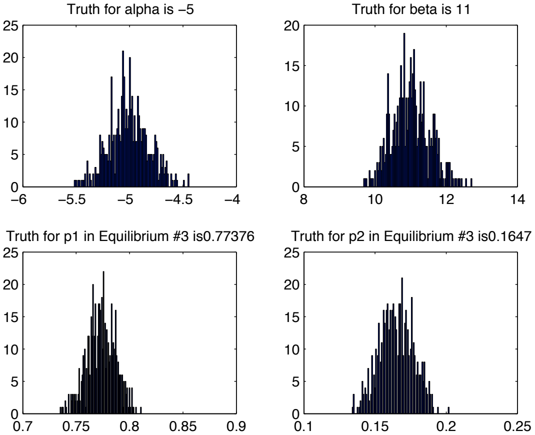
PD Example: Monte Carlo Results with Eq2



PD Example: Monte Carlo Results with Eq1



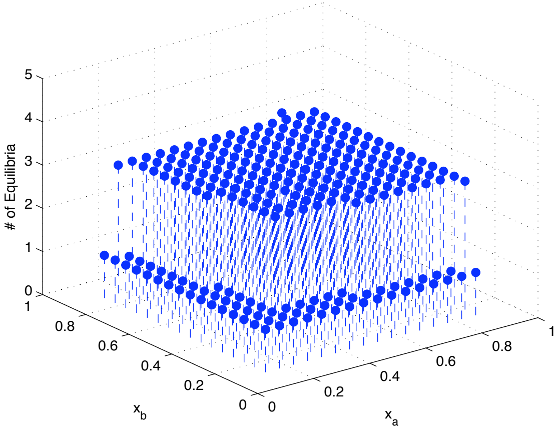
PD Example: Monte Carlo Results with Eq3



PD Example: 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

PD Example: # of Equilibria with Different (x_a^m, x_b^m)

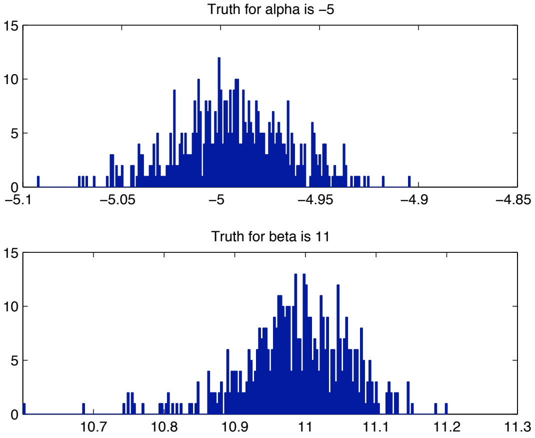


PD Example: Estimation with Multiple Markets

- Constrained optimization formulation for MLE

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

PD Example: Monte Carlo Results with Multiple Markets



2-Step Methods

- Recall the constrained optimization formulation for FIML is

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

- 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: ML

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

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

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

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

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:

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

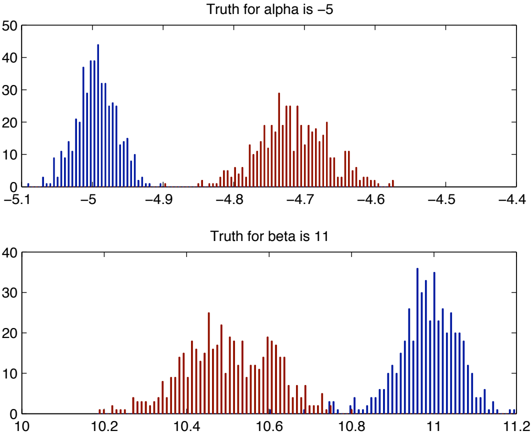
- For dynamic games, Markov perfect equilibrium conditions are characterized by

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

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

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

PD Example: FIML v.s. 2-Step ML

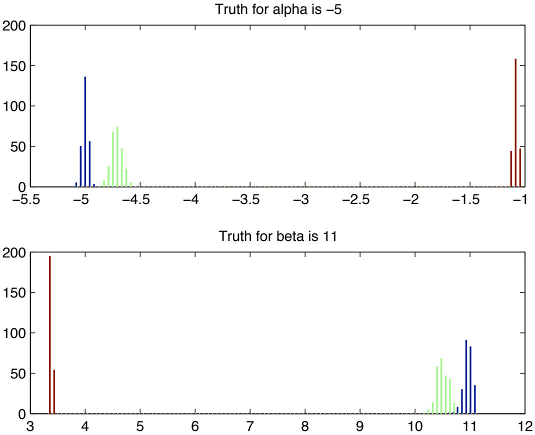


Nested Pseudo Likelihood (NPL): Agguiregabiria and Mira (2007)

- 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
     $$(\alpha^{k+1}, \beta^{k+1}) = \arg\max_{(\alpha,\beta)} \mathcal{L}\left(\Psi_a(\hat{p}_b^k, \alpha, \beta, x_a), \Psi_b(\hat{p}_a^k, \alpha, \beta, x_b), X\right)$$
     2.2 One best-reply iteration on $\hat{p}^k$
     $$\begin{aligned} \hat{p}_a^{k+1} &= \Psi_a(\hat{p}_b^k, \alpha^{k+1}, \beta^{k+1}, x_a) \\ \hat{p}_b^{k+1} &= \Psi_b(\hat{p}_a^k, \alpha^{k+1}, \beta^{k+1}, x_b) \end{aligned}$$
     2.3 Let $k := k + 1$;

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

PD Example: FIML, 2-Step ML and NPL



Experiment 1: Best-Reply Stable Equilibrium with Lowest Probabilities of Confess for Player a in Each Market

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

Estimator	Estimates		RMSE	CPU (sec)	Avg. NPL Iter.
	α	β			
MPEC	-4.999 (0.031)	10.995 (0.062)	0.069	0.94	–
2-Step ML	-4.994 (0.04)	11.002 (0.09)	0.099	0.36	–
2-Step LS	-5.004 (0.04)	11.027 (0.15)	0.159	0.07	–
NPL	-5.001 (0.03)	10.999 (0.065)	0.072	40.26	125

Experiment 2: Best-Reply Stable Equilibrium in Each Market

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

Estimator	Estimates		RMSE	CPU (sec)	Avg. NPL Iter.
	α	β			
MPEC	-5.001 (0.024)	10.994 (0.056)	0.062	1.06	–
2-Step ML	-4.997 (0.03)	11.001 (0.10)	0.108	0.36	–
2-Step LS	-5.007 (0.04)	11.023 (0.17)	0.175	0.06	–
NPL	-5.003 (0.028)	10.996 (0.226)	0.230	41.97	132

Experiment 3: Random Equilibrium in Each Market

- In each market, we randomly choose an equilibrium.

Estimator	Estimates		RMSE	CPU (sec)	Avg. NPL Iter.
	α	β			
MPEC	-4.999 (0.029)	10.999 (0.057)	0.063	1.02	–
2-Step ML	-4.906 (0.04)	10.828 (0.11)	0.231	0.37	–
2-Step LS	-4.767 (0.05)	10.625 (0.16)	0.472	0.06	–
NPL	Not Converged N/A	Not Converged N/A	N/A	152.3	500

Conclusion

- The advances in computational methods (SQP, Interior Point, AD, MPEC) with NLP solvers such as KNITRO, SNOPT, filterSQP, PATH, makes solving structural models tractable and feasible
- User-friendly interfaces (e.g., AMPL, GAMS) makes this as easy to do as Stata, Gauss, and Matlab