Constrained Optimization Approaches to Structural Estimation

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

ZICE 2011
September 3 – 11, 2011

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 Games of Incomplete Information

Part I

Optimization Overview

Unconstrained Optimization: Background

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

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

Optimality conditions: x^* local minimizer:

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

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

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

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

Warning: Any point x that does not satisfy $\|\nabla f(x)\| \leq \tau$ 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

$$\left\{ \begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \text{objective} \\ \text{subject to} & c(x) = 0 & \text{constraints} \\ & x \geq 0 & \text{variables} \end{array} \right.$$

- $f : R^n \to R$, $c : R^n \to R^m$ 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 $X^* z^* = 0 \ \Leftrightarrow\ x_i^* z_i^* = 0$

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 $\nabla^2 \mathcal{L}(x)$

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

Structural Estimation Overview

- Great interest in estimating models based on economic theory
  - Single-agent dynamic decision models: Rust (1987) – NFXP
  - Demand estimation: BLP(1995), Nevo(2000)
  - Static/dynamic games: BBL(2007), Agguiregabiria and Mira (2007)
  - 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

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

- 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

- 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

- 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

AMPL Model: GMCBusExampleMLE.txt

```
#  Define the state space used in the dynamic programming part
param  N;       # number of states used in dynamic programming approximation
set X := 1..N;  # X is the index set of states
param xmin := 0;
param xmax := 100;
param x {i in X} := xmin + (xmax-xmin)/(N-1)*(i-1);  # x[i] denotes state i

#  Define and process the data
param nT;        # number of periods in data
set T := 1..nT;  # T is the vector of time indices
param Xt {T};    # Xt[t] is the true mileage at time t
param dt {T};    # decision at time t

#  The dynamic programming model in the estimation lives on a discrete state
#  Binning process: assign true mileage Xt[t] to the closest state in X
param xt {t in T} := ceil(Xt[t]/(xmax-xmin)*(N-1)+0.5);

#  Define "known" structural parameters
param beta; # discount factor
```

AMPL Model: GMCBusExampleMLE.txt

```
# Define Structural Parameters to be Estimated #
var thetaCost {1..2} >= 0;  # c(x, thetaCost) = thetaCost[1]*x + thetaCost[2]*x^2
var thetaProbs {1..3} >= 0;  # thetaProbs defines Markov chain
var RC >= 0; # Scrap value parameter

#  Define the Markov chain representing the changes in mileage on the x[i] grid.
#  The state increases by some amount in [0,JumpMax]
param JumpRatio;
param JumpMax := (xmax-xmin) * JumpRatio;

#  Define 1st, 2nd, and the end break point for stepwise
# uniform distribution in mileage increase
param M1 := ceil(1/4*JumpMax/(xmax-xmin)*(N-1)+0.5);
param M2 := ceil(3/4*JumpMax/(xmax-xmin)*(N-1)+0.5);
param M := ceil(JumpMax/(xmax-xmin)*(N-1)+0.5);
set Y := 1..M;    #  Y is the vector of elements in transition rule
var TransProb {i in Y} =
if  i <= M1  then  thetaProbs[1]/M1
else if  i > M1 and i <= M2  then  thetaProbs[2]/(M2-M1)
else  thetaProbs[3]/(M-M2);
```

AMPL Model: GMCBusExampleMLE.txt

```
# DECLARE EQUILIBRIUM CONSTRAINT VARIABLES
var EV {X};          # Value Function of each state

# Define auxiliary variables to economize on expressions

# Cost[i] is the cost of regular maintenance at x[i].
var Cost {i in X} = sum {j in 1..2} thetaCost[j]*x[i]^(j);

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

# PayoffDiff[i]  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];

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

AMPL Model: GMCBusExampleMLE.txt

```
# Define objective Likelihood function
# The likelihood function contains two pieces
# First is the likelihood the engine is replaced given time t state in the data.
# Second is the likelihood that the observed transition between t-1
# and t would have occurred

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

AMPL Model: GMCBusExampleMLE.txt

```
# Define the constraints
subject to

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]))* TransProb[j+1];

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]))* TransProb[j+1])
+ (1- sum {k in 0..(N-i-1)} TransProb[k+1])
* log(exp(CbEV[N])+ exp(-RC + CbEV[1]));

Bellman_N:  EV[N] = log(exp(CbEV[N])+ exp(-RC + CbEV[1]));

Probability: sum {i in 1..3} thetaProbs[i] = 1;

EVBound {i in X}: EV[i] <= 50;
```

AMPL Model: GMCBusExampleMLE.txt

```
# Name the problem
problem MPECZurcher:

# Choose the objective function
Likelihood,


# List the variables
EV, RC, thetaCost, thetaProbs,   # Structural parameters & Bellman EQ
TransProb, Cost, CbEV, PayoffDiff, ProbRegMaint,   # Auxiliary variables


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

AMPL/KNITRO Output

```
KNITRO 5.2.0: alg=1
opttol=1.0e-6
feastol=1.0e-6

Problem Characteristics
-----------------------
Objective goal:  Maximize
Number of variables:                     207
    bounded below:                         6
    bounded above:                       201
    bounded below and above:               0
    fixed:                                 0
    free:                                  0
Number of constraints:                   202
    linear equalities:                     1
    nonlinear equalities:                201
    linear inequalities:                   0
    nonlinear inequalities:                0
    range:                                 0
Number of nonzeros in Jacobian:         2785
Number of nonzeros in Hessian:          1620
```

AMPL/KNITRO Output

Iter	Objective	FeasError	OptError	||Step||	CGits
0	-9.932153e+03	9.900e-01			
1	-2.492187e+03	4.345e-01	1.525e+01	2.322e+01	0
2	-2.468145e+03	1.262e-01	6.965e+02	8.265e+00	0
3	-2.438790e+03	1.643e-02	3.474e+02	3.509e+01	0
4	-2.410675e+03	4.867e-02	1.684e+02	1.785e+01	0
5	-2.389049e+03	1.372e-02	8.074e+01	1.097e+01	0
6	-2.371539e+03	8.282e-03	3.682e+01	2.331e+00	0
7	-2.359412e+03	3.551e-03	1.530e+01	2.885e+00	0
8	-2.353608e+03	3.623e-04	5.102e+00	1.238e+00	0
9	-2.352284e+03	1.914e-06	1.005e+00	4.058e-01	0
10	-2.352212e+03	1.523e-08	5.612e-02	1.092e-01	0
11	-2.352211e+03	5.559e-11	1.946e-04	6.817e-03	0
12	-2.352211e+03	1.332e-15	1.000e-08	2.380e-05	0

EXIT: Locally optimal solution found.

AMPL/KNITRO Output

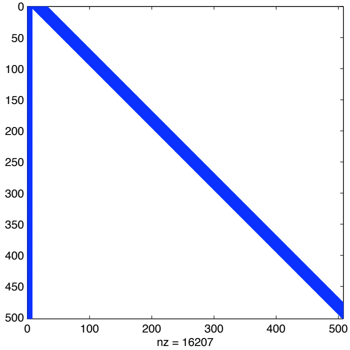
```
Final Statistics
----------------
Final objective value               =  -2.35221126396447e+03
Final feasibility error (abs / rel) =   1.33e-15 / 1.33e-15
Final optimality error  (abs / rel) =   1.00e-08 / 6.71e-10
# of iterations                     =         12
# of CG iterations                  =          0
# of function evaluations           =         13
# of gradient evaluations           =         13
# of Hessian evaluations            =         12
Total program time (secs)           =       0.10326 (     0.097 CPU time)
Time spent in evaluations (secs)    =       0.05323

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

KNITRO 5.2.0: Locally optimal solution.
objective -2352.211264; feasibility error 1.33e-15
12 major iterations; 13 function evaluations
```

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

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

- 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

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

- 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})

Knittel and Metaxoglou (2010)

- 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

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

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

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

- Large sample error in parameter estimates

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

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

$$\min_{(\theta,\xi)} \quad \xi^T Z W Z^T \xi$$
$$\text{subject to} \quad s(\xi,\theta) = S$$

- 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

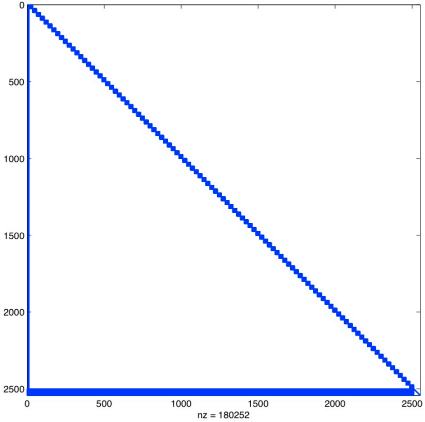
Exploiting Symmetry and Sparsity in the Hessian

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

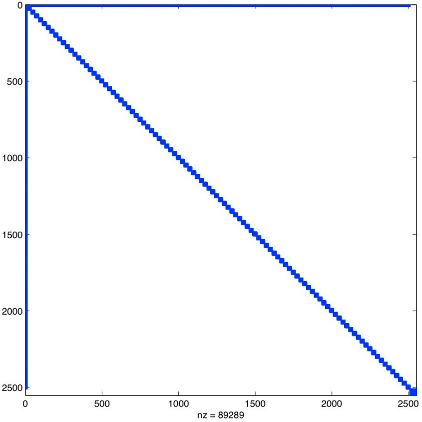
$$\begin{aligned} \min_{(\theta,\xi,g,)} \quad & g^T W g \\ \text{subject to} \quad & s(\delta;\theta_2) = S \\ & Z^T\xi = g \end{aligned}$$

- 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/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
```

Monte Carlo in DFS11: Simulated Data Setup

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

Implementation Details

- MATLAB, highly vectorized code, available at
  `http://faculty.chicagobooth.edu/jean-pierre.dube/vita/MPEC%20code.htm`
- 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 (sec)	Elasticities			Out. Share
				Bias	RMSE	Value	
-2	0.891	NFP	1,300	-0.077	0.14	-10.4	0.91
		MPEC	1,100	-0.076	0.14		
-1	0.928	NFP	1,700	-0.078	0.15	-10.5	0.86
		MPEC	980	-0.077	0.15		
0	0.955	NFP	2,500	-0.079	0.16	-10.6	0.79
		MPEC	910	-0.079	0.16		
1	0.974	NFP	4,300	-0.083	0.16	-10.7	0.69
		MPEC	710	-0.083	0.17		
2	0.986	NFP	6,200	-0.085	0.17	-10.8	0.58
		MPEC	810	-0.085	0.17		
3	0.993	NFP	10,000	-0.088	0.17	-11.0	0.46
		MPEC	640	-0.088	0.17		
4	0.997	NFP	18,000	-0.091	0.16	-11.0	0.35
		MPEC	760	-0.090	0.16		

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 100 starting points or more if you want to find a good solution
- MPEC speed relatively invariant to Lipschitz constant
  - No contraction mapping in MPEC

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	

Speed for Varying # of Markets, Products, Draws

T	J	nn	Lipsch. Const.	Alg	Runs	CPU (sec)	Outside Share
100	25	1000	0.999	NFP MPEC	80% 100%	39,063 1,203	0.45
250	25	1000	0.997	NFP MPEC	100% 100%	80,198 4,174	0.71
500	25	1000	0.998	NFP MPEC	80% 100%	236,244 8,876	0.65
100	25	3000	0.999	NFP MPEC	80% 100%	152,209 3,456	0.46
250	25	3000	0.997	NFP MPEC	100% 100%	288,054 10,771	0.71
25	100	1000	0.993	NFP MPEC	100% 100%	20,413 1,699	0.28
25	250	1000	0.999	NFP MPEC	100% 100%	102,380 8,130	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

$$\begin{aligned} \min_{(\theta,\xi,g,)} \quad & g^T W g \\ \text{subject to} \quad & s(\delta;\theta_2) = S \\ & Z^T \xi = g \end{aligned}$$

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

Part IV

General Formulations

Standard Problem and Current Approach

- Individual solves an optimization problem
- Econometrician observes states and decisions
- Want to estimate structural parameters and equilibrium solutions that are consistent with structural parameters
- Current standard approach
  - Structural parameters: θ
  - Behavior (decision rule, strategy, price): σ
  - Equilibrium (optimality or competitive or Nash) imposes

$$G(\theta, \sigma) = 0$$

  - Likelihood function for data X and parameters θ

$$\max_{\theta} L(\theta; X)$$

  where equilibrium can be presented by $\sigma = \Sigma(\theta)$

NFXP Applied to Single-Agent DP Model – Rust (1987)

- $G(\theta, \sigma) = 0$ represents the Bellman equations
- $\Sigma(\theta)$ is the expected value function and is single-valued as a function of θ
- Outline of NFXP
  - Given θ, compute $\sigma = \Sigma(\theta)$ by solving $G(\theta, \sigma) = 0$
  - For each θ, define

$$L(\theta; X) : \text{likelihood given } \sigma = \Sigma(\theta)$$

  - Solve

$$\max_{\theta} L(\theta; X)$$

NFXP Applied to Random-Coefficeints Demand Estimation – BLP (1995)

- $G(\theta, \sigma) = 0$ represents the demand system of share equations
- $\Sigma(\theta)$ is the unobserved demand shock and is single-valued as a function of θ
- Outline of NFXP
  - Given θ, compute $\sigma = \Sigma(\theta)$ by solving $G(\theta, \sigma) = 0$
  - For each θ, define

$$Q(\theta; X) : \text{GMM given } \sigma = \Sigma(\theta)$$

  - Solve

$$\min_{\theta} Q(\theta; X)$$

NFXP Applied to Games with Multiple Equilibria

- $G(\theta, \sigma) = 0$ characterizes Nash equilibrium for a given θ
- $\Sigma(\theta)$ is set of Nash equilibria given θ and can be multi-valued
- Outline of NFXP
  - Given θ, compute all $\sigma \in \Sigma(\theta)$
  - For each θ, define

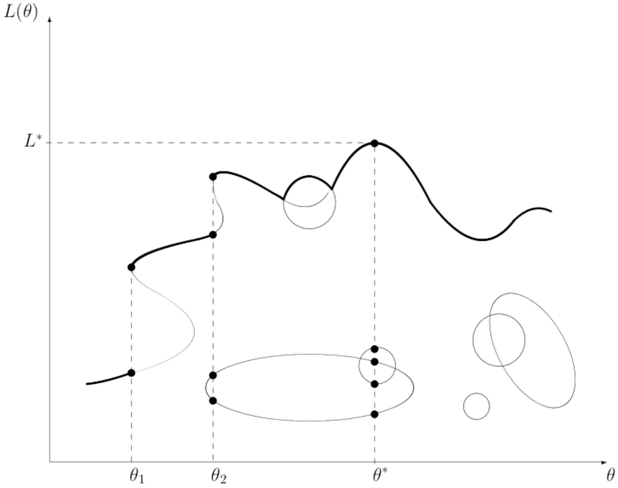
$$L(\theta; X) = \text{max likelihood over all } \sigma \in \Sigma(\theta)$$

  - Solve

$$\max_{\theta} L(\theta; X)$$

- If $\Sigma(\theta)$ is multi-valued, then L can be nondifferentiable and/or discontinuous

NFXP Applied to Games with Multiple Equilibria



NFXP and Related Methods to Games

- NFXP requires finding all σ that solve $G(\theta, \sigma) = 0$, compute the likelihood at each such σ, and report the max as the likelihood value $L(\theta)$
- Finding all equilibria for arbitrary games is an essentially intractable problem - see Judd and Schmedders (2006)
- One fundamental issue: G-S or G-J type methods are often used to solve for an equilibrium. This implicitly imposes an undesired **equilibrium selection rule**: converge only to equilibria that are stable under best reply
- Two-step estimator : Computationally light, very popular, biased in small samples
- NPL – Ag-M(2007): Iterating over the two-step estimator in an attempt to improve the small-sample bias

Constrained Optimization Applied to ML Estimation

- Suppose the game has parameters θ.
- Let σ denote the equilibrium strategy given θ; equilibrium conditions impose

$$G(\theta, \sigma) = 0$$

- For a data set, X,Denote the $augmented\ likelihood$ by $\mathcal{L}(\theta, \sigma; X)$
    - $\mathcal{L}(\theta, \sigma; X)$ decomposes $L(\theta; X)$ so as to highlight the seperate dependence of likelihood on θ and σ
    - In fact, $L(\theta; X) = \mathcal{L}(\theta, \Sigma(\theta); X)$
- Therefore, maximum likelihood estimation is

$$\begin{array}{ll} \max\limits_{(\theta,\sigma)} & \mathcal{L}(\theta, \sigma; X) \\ \text{subject to} & G(\theta, \sigma) = 0 \end{array}$$

Advantanges of Constrained Optimization

- Both $\mathcal{L}$ and G are smooth functions
- Do not require that equilibria are stable under best-reply iteration
- Do not need to solve for all equilibria σ for every θ
- Use multi-start to attempt to find the global solution
- Using a constrained optimization approach allows one to take advantage of the best available methods and software

So ... What is NFXP?

- NFXP is equivalent to nonlinear elimination of variables
- Consider

$$\begin{array}{ll} \max\limits_{(x,y)} & f(x,y) \\ \text{subject to} & g(x,y) = 0 \end{array}$$

  - Define $Y(x)$ implicitly by $g(x, Y(x)) = 0$
  - Solve the unconstrained problem

$$\max_{x} f(x, Y(x))$$

- Used only when memory demands are too large
- Often creates very difficult unconstrained optimization problems

Part V

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

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

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

- Payoff matrix

	$d_a = 1$	$d_a = 0$
$d_b = 1$	$(\theta^a_{11}, \theta^b_{11})$	$(\theta^a_{01}, \theta^b_{01})$
$d_b = 0$	$(\theta^a_{10}, \theta^b_{10})$	$(\theta^a_{00}, \theta^b_{00})$

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

- Utility: Ex-post payoff to prisoners

$$
\begin{aligned}
u_a(d_a, d_b, x_a, \epsilon_a) &= \theta^a_{d_a d_b} x_a + \sigma_a \epsilon_a(d_a) \\
u_b(d_a, d_b, x_b, \epsilon_b) &= \theta^b_{d_a d_b} x_b + \sigma_b \epsilon_b(d_b)
\end{aligned}
$$

  - $(\theta^a_{d_a d_b}, \theta^b_{d_a d_b})$ 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 with 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

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

PD Example with One Market: Solving for Equilibria

- The true values of the structural parameters are

$$
\begin{aligned}
&(\sigma_a, \sigma_b) = (0.1, 0.1) \\
&(\theta_{11}^a, \theta_{11}^b) = (-2, -2) \qquad (\theta_{00}^a, \theta_{00}^b) = (-1, -1) \\
&(\theta_{10}^a, \theta_{01}^b) = (-0.5, -0.5) \qquad (\theta_{01}^a, \theta_{10}^b) = (-0.9, -0.9)
\end{aligned}
$$

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

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

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

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

- 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

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

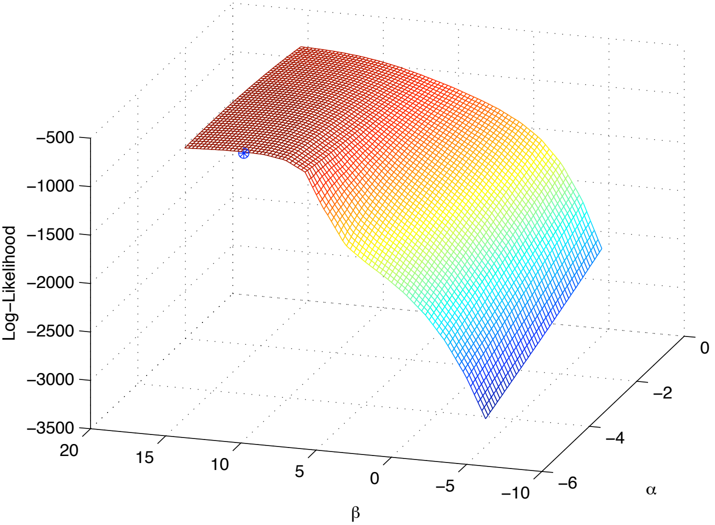
PD Example: MLE via NFXP

- 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:

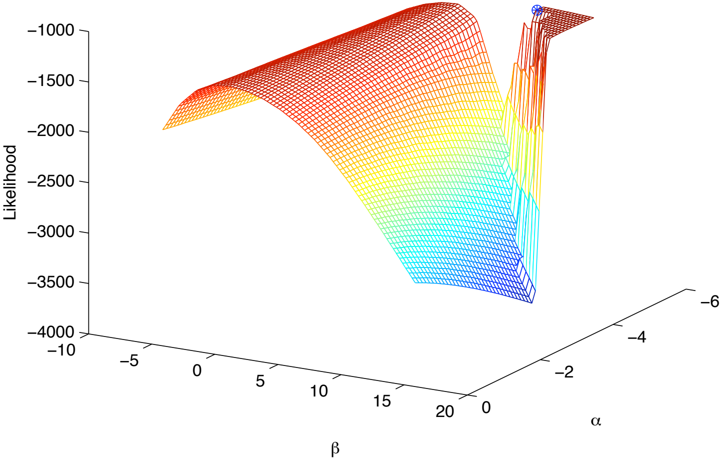
$$k^* = \underset{\{k=1,\ldots,K\}}{\operatorname{argmax}} \, log\mathcal{L}\left(p_a^k(\alpha, \beta), p_b^k(\alpha, \beta); X\right)$$

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

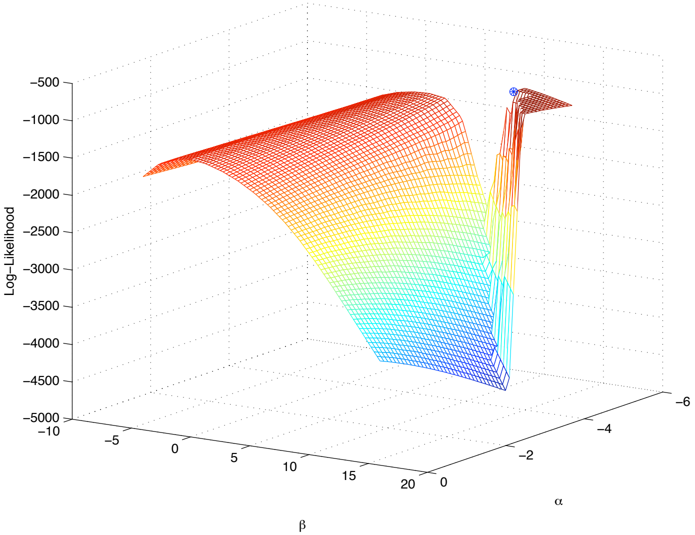
PD Example: Likelihood as a Function of (α, β) – Eq 1



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

$$\max_{(\alpha,\beta,p_a,p_b)} \quad log\mathcal{L}\,(p_a, p_b; X)$$

$$= \sum_{i=1}^{1000} \left(d_a^i * \log(p_a) + (1 - d_a^i) * \log(1 - p_a)\right)$$

$$+ \sum_{i=1}^{1000} \left(d_b^i * \log(p_b) + (1 - d_b^i) * \log(1 - p_b)\right)$$

subject to

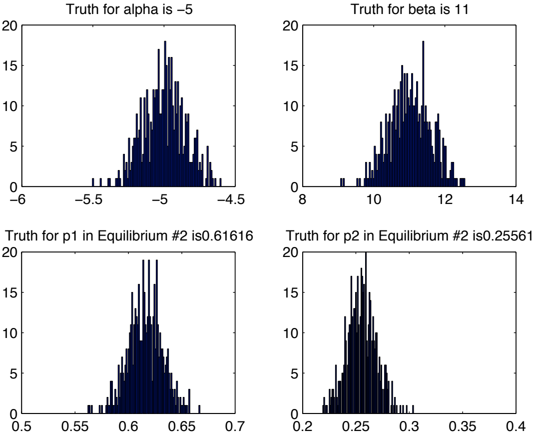
$$p_a = \frac{1}{1 + exp\{0.52(\alpha) + p_b 0.52(\beta - \alpha)\}}$$

$$p_b = \frac{1}{1 + exp\{0.22(\alpha) + p_a 0.22(\beta - \alpha)\}}$$

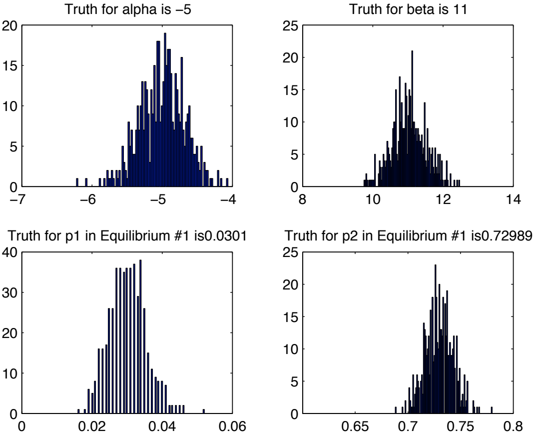
$$0 \leq p_a, p_b \leq 1$$

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

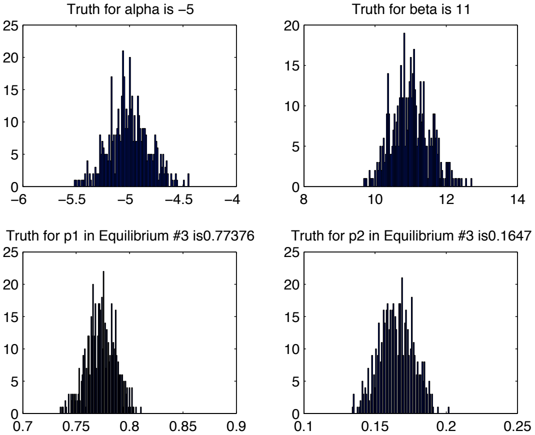
PD Example: Monte Carlo Results with Eq2



PD Example: Monte Carlo Results with Eq1



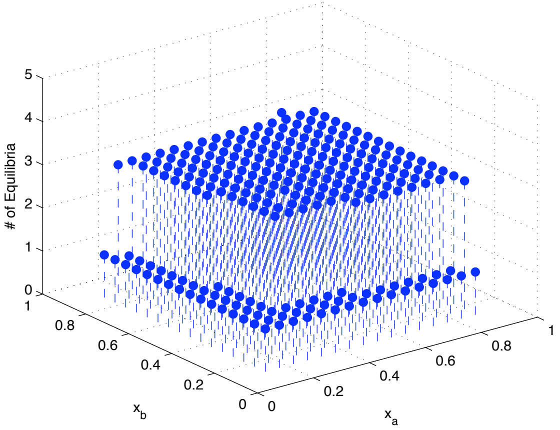
PD Example: Monte Carlo Results with Eq3



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)

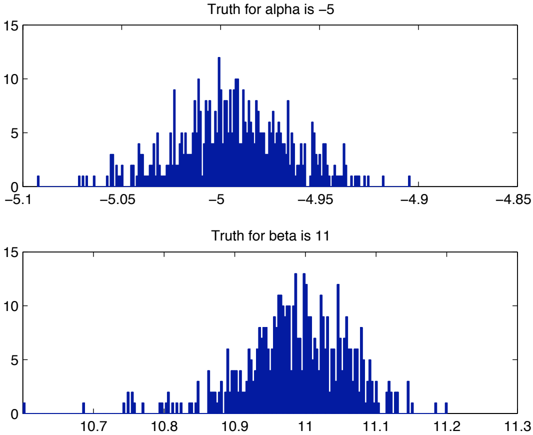


PD Example: Estimation with Multiple Markets

- Constrained optimization formulation for MLE

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

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:

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

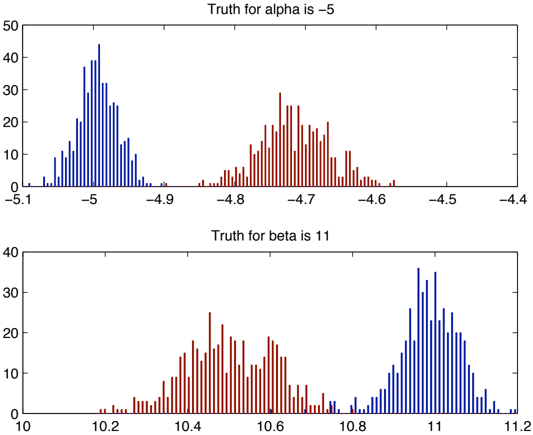
- For dynamic games, Markov perfect equilibrium conditions are characterized by

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

- 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

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

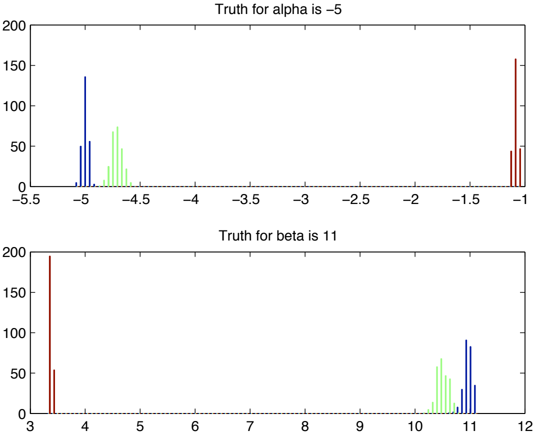
     2.2 One best-reply iteration on $\hat{p}^k$

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

     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



DGP 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)	0.688	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

DGP 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

DGP 3: Random Equilibrium in Each Market

- In each market, we randomly choose an equilibrium.

Estimator	Estimates		RMSE	CPU	Avg. NPL
	α	β		(sec)	Iter.
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	300

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