Constrained Optimization Approaches to Structural Estimation

Che-Lin Su
CMS-EMS
Kellogg School of Management
Northwestern University
c-su@kellogg.northwestern.edu

Institute for Computational Economics
The University of Chicago
July 31- August 9, 2007

Outline of Three Lectures

1. Introduction to Structural Estimation

Outline of Three Lectures

1. Introduction to Structural Estimation
2. Estimation of Demand Systems

Outline of Three Lectures

1. Introduction to Structural Estimation
2. Estimation of Demand Systems
3. Estimation of Dynamic Programming Models of Individual Behavior

Outline of Three Lectures

1. Introduction to Structural Estimation
2. Estimation of Demand Systems
3. Estimation of Dynamic Programming Models of Individual Behavior
4. Estimation of Games

Part I

Introduction, Applications, Example, and Formulations

Structural Estimation

- Great interest in estimating models based on economic structure
  - Demand Estimation: BLP(1995), Nevo(2000)
  - Dynamic programming models of individual behavior: Rust (1987)
  - Nash equilibria of games – static, dynamic
  - Dynamic stochastic general equilibrium
- General belief: Estimation is a major computational challenge because it involves solving model many times
- Our goal: Teach you more computational efficient ways of estimating structural models
- Our finding: Many supposed computational "difficulties" can be avoided by using optimization tools developed in numerical analysis over the past 40 years

Current Views on Structural Estimation

- Erdem et al. (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.*

Simple Consumer Demand Example

- Data and Model
  - Data on demand, q, and price p, but demand is observed with error ε
  - True demand is $q - \varepsilon$
  - Assume a parametric form for utility function $u(c;\beta)$ where β is a vector of parameters
  - Economic theory implies

$$u_c(c;\beta) = u_c(q-\varepsilon;\beta) = p$$

- Standard Approach
  - Assume, for example, a functional form for utility $u(c) = c - \beta c^2$
  - Solve for demand function $c = (1-p)/(2\beta)$
  - Hence, for some ε_i, the i'th data point satisfies

$$q_i = (1-p_i)/(2\beta) + \varepsilon_i$$

  - Choose β to minimize the sum of squared errors

$$\sum_{i=1} \left(q_i - (1-p_i)/(2\beta)\right)^2$$

Simple Consumer Demand Example

- Limitations
  - Need to solve for demand function, which is hard if not impossible
  - For example, suppose

$$u(c) = c - \beta\left(c^2 + \gamma c^4 + \delta c^6\right)$$

    with first-order condition

$$1 - \beta\left(2c + 4\gamma c^3 + 6\delta c^5\right) = p$$

  - There is no closed-form solution for demand function!
  - What were you taught to do in this case? *Change the model*!
- Proper Procedure
  - Deal with the first-order condition directly since it has all the information you can have.
  - Recognize that all you do is find the errors that minimize their sum of squares and are consistent with structural equations.

Simple Consumer Demand Example

- For our consumption demand model, this is the problem

$$\begin{aligned} \min_{\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{subject to} \quad & u_c\left(q_i - \varepsilon_i; \beta\right) = p_i, \quad \forall i \quad \text{[F.O.C.]} \end{aligned}$$

  - In the case of the quadratic utility function, this reduces to

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{subject to} \quad & 1 - 2\beta c_i = p_i, \quad \forall i \quad \text{[F.O.C.]} \\ & q_i = c_i + \varepsilon_i \end{aligned}$$

  - Degree-six utility function with MPEC approach solves

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{subject to} \quad & 1 - \beta\left(2c_i + 4\gamma c_i^3 + 6\delta c_i^5\right) = p_i, \quad \forall i \quad \text{[F.O.C.]} \\ & q_i = c_i + \varepsilon_i \end{aligned}$$

- Big problem but constraint Jacobian is sparse

Simple Consumer Demand Example

- Even when you can solve for demand function, you may not want to
  - Consider the case

$$\begin{aligned} u(c) &= c - \beta_1 c^2 - \beta_2 c^3 - \beta_3 c^4 \\ u'(c) &= 1 - 2\beta_1 c - 3\beta_2 c^2 - 4\beta_3 c^3 \end{aligned}$$

  - Demand function is

$$\begin{aligned} q &= \frac{1}{12\beta_3} W - \frac{1}{4} \frac{8\beta_1\beta_3 - 3\beta_2^2}{\beta_3 W} - \frac{1}{4}\frac{\beta_2}{\beta_3} \\ W &= \sqrt[3]{\left(108\beta_1\beta_2\beta_3 - 216\beta_3^2 p + 216\beta_3^2 - 27\beta_2^3 + 12\sqrt{3}\beta_3 Z\right)} \\ Z &= \sqrt{Z_1 + Z_2} \\ Z_1 &= 32\beta_1^3\beta_3 - 9\beta_1^2\beta_2^2 - 108\beta_1\beta_2\beta_3 p + 108\beta_1\beta_2\beta_3 \\ Z_2 &= 108\beta_3^2 p^2 - 216\beta_3^2 p + 27 p\beta_2^3 + 108\beta_3^2 - 27\beta_2^3 \end{aligned}$$

  - Demand function is far costlier to compute than the first-order conditions
- The (*bad*) habit of restricting models to cases with closed-form solutions is completely unnecessary.

Summary

- Constrained optimization formulation for the consumption demand model is

$$\begin{aligned} \min_{c_i, \varepsilon_i, \beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{subject to} \quad & u_c\left(c_i; \beta\right) = p_i, \quad \forall i \quad \text{[F.O.C.]} \\ & q_i = c_i + \varepsilon_i \end{aligned}$$

- The problem is big; need one variable and one constraint per price-quantity observation
- NLP solvers work fine despite large number of variables and constraints. It is because the constraint Jacobian matrix is sparse

Part II

Random-Coefficients Demand Estimation

Random-Coefficients Logit Models of Demand

- Setup:
  - Market $t = 1, \ldots, T$
  - Product $j = 1, \ldots, J$ in each market
  - Consumer $i = 1, \ldots, I_t$ in market t
- For each market, we observe aggregate quantities, average prices, and product characteristics
- utility of consumer i for product j in market t

$$u_{ijt} = \alpha_i(y_i - p_{jt}) + x_{jt}\beta_i + \xi_{jt} + \epsilon_{ijt}$$

  - y_i: income of consumer i
  - p_{jt}: price of product j in market t
  - x_{jt}: a vector of observable characteristics of product j
  - ξ_{jt}: unobserved (by the econometrician) product characteristics
  - ϵ_{ijt}: mean-zero stochastic term
  - α_i, β_i: individual-specific taste coefficients to be estimated

Random-Coefficients Logit Models of Demand

- A little bit more details on α_i and β_i:

$$\begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \Pi D_i + \Sigma \nu_i, \qquad \nu_i \sim P_\nu^*(\nu), \quad D_i \sim \hat{P}_D^*(D)$$

  - D_i: unobserved demographic variables for consumer i
  - ν_i: additional unobserved individual variables
  - α, β: mean of taste coefficients
  - Π, Σ: measure of how taste characteristics vary with D_i and ν_i

- Structural parameters to be estimated $\theta = (\theta_1, \theta_2)$:

$$\theta_1 = (\alpha, \beta)$$
$$\theta_2 = (\Pi, \Sigma)$$

Random-Coefficients Logit Models of Demand

- Back to the utility u_{ijt}

$$\begin{aligned} u_{ijt} &= \alpha_i y_i + \delta_{jt}(x_{jt}, p_{jt}, \xi_{jt}; \theta_1) + \mu_{ijt}(x_{jt}, p_{jt}, \mu_i, D_i; \theta_2) + \epsilon_{ijt} \\ \delta_{jt} &= x_{jt}\beta - \alpha p_{jt} + \xi_{jt} \\ \mu_{ijt} &= [-p_{jt}, x_{jt}](\Pi D_i + \Sigma \nu_i) \end{aligned}$$

  - δ_{jt}: mean utility for all consumers for product j in market i
  - $\mu_{ijt} + \epsilon_{ijt}$: deviation from the mean utility δ_{jt}
- Consumer i purchases product j^* in market t if

$$u_{ij^*t} \geq u_{ijt}, \quad \forall j = 1, \ldots, J$$

- Market shares equations: $s_{jt}(\delta, \theta_2) = S_{jt}, \quad \forall j, t$
  - S_{jt}: observed market share of product j in market t
  - $s_{jt}(\delta, \theta_2)$: estimated market share

BLP Estimation Algorithm

- Outer loop: $\min_{\theta} \omega(\theta)^T Z \Phi^{-1} Z^T \omega(\theta)$
  - Guess $\theta = (\theta_1, \theta_2)$ parameters to compute $\omega(\theta)$:

$$\omega_{jt}(\theta) = \delta_{jt}(S_{\cdot t}; \theta_2) - (x_{jt}\beta + \alpha p_{jt}) \equiv \xi_{jt}$$

  - $\theta_1 = (\alpha, \beta)$; $\theta_2 = (\Pi, \Sigma)$
  - ξ_{jt}: unobserved product characteristic

BLP Estimation Algorithm

- Outer loop: $\min_{\theta} \omega(\theta)^T Z \Phi^{-1} Z^T \omega(\theta)$
  - Guess $\theta = (\theta_1, \theta_2)$ parameters to compute $\omega(\theta)$:

$$\omega_{jt}(\theta) = \delta_{jt}(S_{\cdot t}; \theta_2) - (x_{jt}\beta + \alpha p_{jt}) \equiv \xi_{jt}$$

  - $\theta_1 = (\alpha, \beta)$; $\theta_2 = (\Pi, \Sigma)$
  - ξ_{jt}: unobserved product characteristic
- Inner loop: compute mean utility δ for a given θ_2
  - Solve $s_{\cdot t}(\delta; \theta_2) = S_{\cdot t}$ by contraction mapping to get δ:

$$\delta_{\cdot t}^{h+1} = \delta_{\cdot t}^{h} + ln\, S_{\cdot t} - ln\, s(\delta_{\cdot t}^{h}; \theta_2)$$

  - $s_{jt}(\delta; \theta_2) = \frac{1}{ns} \sum_{i=1}^{ns} \frac{\exp[\delta_{jt} + \mu_{ijt}]}{1 + \sum_{m=1}^{J} \exp[\delta_{mt} + \mu_{imt}]}$

BLP Estimation Algorithm

- Outer loop: $\min_{\theta} \omega(\theta)^T Z \Phi^{-1} Z^T \omega(\theta)$
  - Guess $\theta = (\theta_1, \theta_2)$ parameters to compute $\omega(\theta)$:

$$\omega_{jt}(\theta) = \delta_{jt}(S_{\cdot t}; \theta_2) - (x_{jt}\beta + \alpha p_{jt}) \equiv \xi_{jt}$$

  - $\theta_1 = (\alpha, \beta)$; $\theta_2 = (\Pi, \Sigma)$
  - ξ_{jt}: unobserved product characteristic
- Inner loop: compute mean utility δ for a given θ_2
  - Solve $s_{\cdot t}(\delta; \theta_2) = S_{\cdot t}$ by contraction mapping to get δ:

$$\delta_{\cdot t}^{h+1} = \delta_{\cdot t}^{h} + ln\, S_{\cdot t} - ln\, s(\delta_{\cdot t}^{h}; \theta_2)$$

  - $s_{jt}(\delta; \theta_2) = \frac{1}{ns} \sum_{i=1}^{ns} \frac{\exp[\delta_{jt} + \mu_{ijt}]}{1 + \sum_{m=1}^{J} \exp[\delta_{mt} + \mu_{imt}]}$
- Stopping rules: need very high accuracy (in relative error) from the inner loop in order for the outer loop to converge

Weaknesses of BLP

- Contraction mapping is linear convergent at best
  - Numerical performance will be sensitive to the stopping rules
- Stopping rules
  - needs very high accuracy (in relative error) from the inner loop in order for the outer loop to converge
  - needs to be careful at setting tolerance levels for both inner and outer loop
  - Theory: outer loop accuracy $= \sqrt{\text{inner loop accuracy}}$
  - If the inner loop has 8 digits accuracy (10^{-8}), then the outer loop has 4 digits accuracy ($\sqrt{10^{-8}} = 10^{-4}$)

MPEC Applied to BLP

- Fox and Su (2007): Improving the Numerical Performance of BLP Structural Demand Estimators
- Constrained optimization formulation

$$\begin{aligned} \min_{(\theta,\delta,\omega)} \quad & \omega^T Z \Phi^{-1} Z^T \omega \\ \text{s.t.} \quad & \omega = \delta - (x\beta + \alpha p) \\ & s(\delta; \theta_2) = S \end{aligned}$$

- Advantages:
  - Faster
  - Fewer iterations/function evaluations
  - Easy to code in AMPL and to access good NLP solvers
  - No need to worry about setting up tolerance levels
- Notes: Jacobian is dense; size is 2300 by 2300, but AD saves us

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]) ;
```

Numerical Performance

We consider the example in Nevo (2000)

	MPEC	BLP
Implementation	AMPL/Knitro(Interior Point)	Matlab
Timing	$\sim$13 sec.	$\sim$66 sec.
# of θ visited	7	122
Function Evaluations	8	$>$ 2000
Objective Value	4.65	4.65

Summary

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

$$\begin{aligned} \min_{(\theta,\delta,\omega)} \quad & \omega^T Z \Phi^{-1} Z^T \omega \\ \text{s.t.} \quad & \omega = \delta - (x\beta + \alpha p) \\ & s(\delta; \theta_2) = S \end{aligned}$$

- MPEC approach with AMPL/KNITRO works very well
- The constraint Jacobian is dense and the computer memory requirement is high
- The memory problems are cheaply solved (RAM is cheap) !!!

Standard Problem and Current Approach

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

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

  - Likelihood function for data X and parameters θ

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

where equilibrium can be represented by a function $\sigma = \Sigma(\theta)$

MPEC Ideas Applied to Estimation

- Suppose that an economic model has parameters θ
  - Suppose that equilibrium and optimality imply that the observable economic variables, x, follow a stochastic process parameterized by a finite vector σ
  - The value of σ will depend on θ through a set of equilibrium conditions
$$0 = G(\theta, \sigma)$$
  - Denote the augmented likelihood of a data set, X, by $\mathcal{L}(\theta, \sigma; X)$
- Therefore, maximum likelihood is the constrained optimization problem

$$\begin{aligned} \max_{(\theta,\sigma)} \quad & \mathcal{L}(\theta, \sigma; X) \\ \text{subject to} \quad & 0 = G(\theta, \sigma) \end{aligned}$$

Our Advantanges

- We do not require that equilibrium be defined as a solution to a fixed-point equation.
- We do not need to specify an algorithm for computing σ given θ; good solvers will probably do better.
- Gauss-Jacobi or Gauss-Seidel methods are often used in economics even though they are at best linearly convergent, whereas good solvers are at least superlinearly convergent locally (if not much better) and have better global properties than GJ and GS typically do.
- Using a direct optimization approach allows one to take advantage of the best available methods and software (AMPL, KNITRO, SNOPT, filterSQP, PATH, etc) from the numerical analysis

Constrained Estimation

- The MPEC approach is an example of constrained estimation, be it maximum likelihood or method of moments.
- Sampling of previous literature
  - Aitchison, J. & S.D. Silvey. Maximum Likelihood Estimation of Parameters Subject to Restraints. *Annals of Mathematica1 Statistics* 29 (1958): 813-828.
  - Gallant, A. Ronald, and Alberto Holly. Statistical Inference in an Implicit, Nonlinear, Simultaneous Equation Model in the Context of Maximum Likelihood Estimation. *Econometrica*, Vol. 48, No. 3 (April, 1980)
  - Gallant, A. Ronald, and George Tauchen. Seminonparametric Estimation of Conditionally Constrained Heterogeneous Processes: Asset Pricing Applications. *Econometrica*, Vol. 57, No. 5 (Sep., 1989), pp. 1091-1120.
  - Silvey, S. D. *Statistical Inference*. London: Chapman & Hall, 1970.
  - Wolak, F. A. An Exact Test for Multiple Inequality and Equality Constraints in the Linear Regression Model. *J. Am. Statist. Assoc.* 82 (1987): 782-93.
  - Wolak, F.A. Testing inequality constraints in linear econometric models. *Journal of Econometrics*, 1989.

Part III

Estimation of Dynamic Programming Models

Canonical Example - Dynamic Programming

- Individual solves a dynamic programming problem
- Econometrician observes state and decisions
- Augmented likelihood function for data X

$$\mathcal{L}(\theta, \sigma; X)$$

  where θ is set of parameters and σ is decision rule
- Rationality imposes a relationship between θ and σ

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

- We want to find maximum likelihood θ but impose rationality condition

Zurcher's Bus Engine Replacement Problem

- 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

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

  - Repairman has temporary shock to ordinary maintenance cost
  - Repairman solves DP:

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

- Econometrician
  - Observes mileage and decision, but not cost
  - Assumes extreme value distribution
- Structural parameters to be estimated
  - Coefficients of maintenance cost function; e.g., $c(x, \theta^c) = \theta_1^c x + \theta_2^c x^2$
  - Overhaul cost RC
  - Transition probabilities in mileages $\theta^p(x_{t+1} - x_t)$

Zurcher Model – 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

Nested Fixed Point Algo: Rust (Econometrica, 1987)

- Outer loop: Solve likelihood

$$\max_{\theta \geq 0} L\left(\theta, EV_\theta; X\right)$$

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

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

  - In practice, this means writing a program for EV_θ
  - Rust started with contraction iterations and then switched to Newton iterations
- Problem with NFXP: Must compute EV_θ to high accuracy for each θ examined
  (i) for outer loop to converge; see the BLP slide
  (ii) to obtain accurate numerical derivatives for the outer loop

MPEC Approach for Solving Zucher Model

- MPEC
  - Form augmented likelihood function for data X

$$\mathcal{L}(\theta, EV; X)$$

    where θ is set of parameters and EV is the value function
  - Rationality and Bellman equation imposes a relationship between θ and EV

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

  - Solve constrained optimization problem

$$\begin{aligned} \max_{(\theta, EV)} \quad & \mathcal{L}(\theta, EV; X) \\ \text{s.t.} \quad & EV = T(EV, \theta) \end{aligned}$$

AMPL Model: MPEC_RustBus.mod

```
#####  HAROLD ZURCHER BUS REPAIR EXAMPLE  #####
#
# An MPEC Approach to compute maximum likelihood estimates of the
# Harold Zurcher bus problem in Rust, Econometrica, 1987.
# Ken Judd and Che-Lin Su,
# February 21, 2006. Current version: October 26, 2006
#
#####  SET UP THE PROBLEM  #####
#
#  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
#  x[i] denotes state i; the set of states is a uniform grid on the interval [xmin, xmax]
param xmin := 0;
param xmax := 100;
param x {i in X} := xmin + (xmax-xmin)/(N-1)*(i-1);
#
#  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
#  We fix beta since data cannot identify it
param beta;       # discount factor
```

AMPL Model: MPEC_RustBus.mod

```
#####  DECLARE STRUCTURAL PARAMETERS TO BE ESTIMATED #####
#####  Parameters for cost function  #####
#  c(x, thetaCost) = thetaCost[1]*x + thetaCost[2]*x^2
var thetaCost {1..2} >= 0;
#####  Parameters and definition of transition process  #####
 #  thetaProbs defines Markov chain transition probabilities
var thetaProbs {1..3} >= 0.00001;

#  Define the Markov chain representing the changes in mileage on the x[i] grid.
#  The state increases by some amount in [0,JumpMax] where
#  JumpMax is the maximum increase in mileage in one period
#  and equals a fraction JumpRatio of the range of mileage in the state space
param JumpRatio;
param JumpMax := (xmax-xmin) * JumpRatio;
#  We assume that the jumps are independent of the current state
#  and its transition process has three pieces, each a uniform distribution
#  Define 1st break point for stepwise uniform distribution in mileage increase
param M1 := ceil(1/4*JumpMax/(xmax-xmin)*(N-1)+0.5);
#  Define 2nd break point for stepwise uniform distribution in mileage increase
param M2 := ceil(3/4*JumpMax/(xmax-xmin)*(N-1)+0.5);
#  Define end point for stepwise uniform distribution in mileage increase
param M := ceil(JumpMax/(xmax-xmin)*(N-1)+0.5);
#  Y is the vector of elements in transition rule
set Y := 1..M;
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);
#####  Scrap value parameter  #####
var RC >= 0;
##### END OF STRUCTURAL VARIABLES  #####
```

AMPL Model: MPEC_RustBus.mod

```
#####  DECLARE EQUILIBRIUM CONSTRAINT VARIABLES #####
# The NLP approach requires us to solve equilibrium constraint variables

var EV {X};          # Value Function of each state

#####  END OF EQUILIBRIUM CONSTRAINT VARIABLES  #####
#####  DECLARE AUXILIARY VARIABLES  ######
#  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} = sum {j in 1..2} thetaCost[j]*x[i]^(j);

#  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]));
var BellmanViola {i in 1..(N-M+1)} = sum {j in 0..(M-1)} log(exp(CbEV[i+j])
                                    +exp(-RC + CbEV[1]))* TransProb[j+1] - EV[i];

#####  END OF AUXILIARY VARIABLES  #####
```

AMPL Model: MPEC_RustBus.mod

```
#####  OBJECTIVE AND CONSTRAINT DEFINITIONS  #####

## Define objective function:  Likelihood function
maximize Likelihood:
## The likelihood function contains two pieces
## First is the likelihood that the engine is replaced given time t state in the data.
sum {t in 2..nT} log(dt[t]*(1-ProbRegMaint[xt[t]]) + (1-dt[t])*ProbRegMaint[xt[t]])
## Second is the likelihood that the observed transition between t-1 and t would have occurred.
+ 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]));

#  List the constraints

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

Probability: sum {i in 1..3} thetaProbs[i] = 1;  #  The probability in transition process add to one

#  Put bound on EV; this should not bind, but is a cautionary step to help keep algorithm within bounds

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

AMPL Model: MPEC_RustBus.mod

```
####   DEFINE THE PROBLEM   #####

# Name the problem

problem MPECZurcher:

# Choose the objective function

Likelihood,

# List the variables

EV, RC, thetaCost, thetaProbs, TransProb, Cost, CbEV, PayoffDiff, ProbRegMaint, BellmanViola,

# List the constraints

Bellman_1toNminusM,
Bellman_LastM,
Bellman_N,
Probability,
EVBound;

#################################
```

AMPL Command: MPEC_RustBus.cmd

```
# Call solver SNOPT and give it options

option snopt_options $snopt_options 'outlev=2 timing=1 ';  # output level

# Initial guesses set at trivial values; probably not good initial guess
     let {i in X} EV[i] := 0;
     let {i in 1..3} thetaProbs[i] := 1/3;


# Solve command
     solve MPECZurcher;

display _solve_time;

# Output commands
     option display_round 6, display_width 120;

# write the value function
     display EV;

# write the structural parameters (remember beta was fixed)
     display beta, RC, thetaCost, thetaProbs, TransProb;

# write errors in Bellman equations
      display Bellman_1toNminusM.body;
      display Bellman_LastM.body;
      display Bellman_N.body;
      display BellmanViola;
```

MPEC Applied to Zucher: Three-Parameter Estimates

- Synthetic data is better: avoids misspecification
- Use Rust's estimates to generate 2 synthetic data sets of 10^3 and 10^4 data points respectively.
- Timing for estimating three parameters (as in the Rust)
- AMPL program solved on NEOS server using SNOPT

T	N	Estimates RC	θ_1^c	θ_2^c	CPU (sec)	Major Iterations	Evals*	Bell. EQ. Error
10^3	101	1.112	0.043	0.0029	0.14	66	72	3.0E−13
10^3	201	1.140	0.055	0.0015	0.31	44	59	2.9E−13
10^3	501	1.130	0.050	0.0019	1.65	58	68	1.4E−12
10^3	1001	1.144	0.056	0.0013	5.54	58	94	2.5E−13
10^4	101	1.236	0.056	0.0015	0.24	59	67	2.9E−13
10^4	201	1.257	0.060	0.0010	0.44	59	67	1.8E−12
10^4	501	1.252	0.058	0.0012	0.88	35	45	2.9E−13
10^4	1001	1.256	0.060	0.0010	1.26	39	52	3.0E−13

*Number of function and constraint evaluations

MPEC Applied to Zucher: Five-Parameter Estimates

- Rust did a two-stage procedure, estimating transition parameters in first stage. We do full ML

		Estimates					CPU	Maj.	Evals	Bell.
T	N	RC	θ_1^c	θ_2^c	θ_1^p	θ_2^p	(sec)	Iter.		Err.
10^3	101	1.11	0.039	0.0030	0.723	0.262	0.50	111	137	6E−12
10^3	201	1.14	0.055	0.0015	0.364	0.600	1.14	109	120	1E−09
10^3	501	1.13	0.050	0.0019	0.339	0.612	3.39	115	127	3E−11
10^3	1001	1.14	0.056	0.0014	0.360	0.608	7.56	84	116	5E−12
10^4	101	1.24	0.052	0.0016	0.694	0.284	0.50	76	91	5E−11
10^4	201	1.26	0.060	0.0010	0.367	0.053	0.86	85	97	4E−11
10^4	501	1.25	0.058	0.0012	0.349	0.596	2.73	83	98	3E−10
10^4	1001	1.26	0.060	0.0010	0.370	0.586	19.12	166	182	3E−10

Observations

- Problem is solved very quickly.
- Timing is nearly linear in the number of states for modest grid size.
- The likelihood function, the constraints, and their derivatives are evaluated only 45-200 times in this example.
- In contrast, the Bellman operator in NFXP (the constraints here) is evaluated hundreds of times in NFXP

Parametric Bootstrap Experiment

- For calculating statistical inference, bootstrapping is better and more reliable than asymptotic analysis. However, bootstrap is often viewed as computationally infeasible
- Examine several data sets to determine patterns
- Use Rust's estimates to generate 1 synthetic data set
- Use the estimated values on the synthetic data set to reproduce 20 independent data sets:
  - Five parameter estimation
  - 1000 data points
  - 201 grid points in DP

Maximum Likelihood Parametric Bootstrap Estimates

Table 3: Maximum Likelihood Parametric Bootstrap Results

	Estimates						CPU	Maj.	Evals	Bell.
	RC	θ_1^c	θ_2^c	θ_1^p	θ_2^p	θ_3^p	(sec)	Ite		Err.
mean	1.14	0.037	0.004	0.384	0.587	0.029	0.54	90	109	8E−09
S.E.	0.15	0.035	0.004	0.013	0.012	0.005	0.16	24	37	2E−08
Min	0.95	0.000	0.000	0.355	0.571	0.021	0.24	45	59	1E−13
Max	1.46	0.108	0.012	0.403	0.606	0.039	0.88	152	230	6E−08

MPEC Approach to Method of Moments

- Suppose you want to fit moments. E.g., likelihood may not exist
- Method then is

$$\begin{aligned} \min_{(\theta,\sigma)} \quad & \|m(\theta,\sigma) - M(X)\|^2 \\ \text{s.t.} \quad & G(\theta,\sigma) = 0 \end{aligned}$$

  - Compute moments $m(\theta,\sigma)$ numerically via linear equations in constraints - no simulation
  - Objective function for the Rust's bus example:

$$\begin{aligned} \mathcal{M}(m,M) \quad = & (m_x - M_x)^2 + (m_d - M_d)^2 + (m_{xx} - M_{xx})^2 + (m_{xd} - M_{xd})^2 \\ & + (m_{dd} - M_{dd})^2 + (m_{xxx} - M_{xxx})^2 + (m_{xxd} - M_{xxd})^2 \\ & + (m_{xdd} - M_{xdd})^2 + (m_{ddd} - M_{ddd})^2 \end{aligned}$$

Formulation for Method of Moments

- Constraints imposing equilibrium conditions and moment definitions, and computes stationary distribution p

$$
\begin{aligned}
\max_{(\theta,\sigma,\Pi,p,m)} \quad & \mathcal{M}(m, M) \\
\text{s.t.} \quad & G(\theta, \sigma) = 0, \quad \Pi = H(\theta, \sigma) \\
& p^\top \Pi = p^\top, \quad \sum_{x \in Z, d \in \{0,1\}} p_{x,d} = 1 \\
& m_x = \sum_{x,d} p_{x,d}\, x, \;\; m_d = \sum_{x,d} p_{x,d}\, d \\
& m_{xx} = \sum_{x,d} p_{x,d}\, (x - m_x)^2, \;\; m_{xd} = \sum_{x,d} p_{x,d}\, (x - m_x)(d - m_d) \\
& m_{dd} = \sum_{x,d} p_{x,d}\, (d - m_d)^2 \\
& m_{xxx} = \sum_{x,d} p_{x,d}\, (x - m_x)^3, \;\; m_{xxd} = \sum_{x,d} p_{x,d}\, (x - m_x)^2 (d - m_d) \\
& m_{xdd} = \sum_{x,d} p_{x,d}\, (x - m_x)(d - m_d)^2, \;\; m_{ddd} = \sum_{x,d} p_{x,d}\, (d - m_d)^3
\end{aligned}
$$

Method of Moments Parametric Bootstrap Estimates

Table 4: Method of Moments Parametric Bootstrap Results

	Estimates						CPU	Major	Evals	Bell
	RC	θ_1^c	θ_2^c	θ_1^p	θ_2^p	θ_3^p	(sec)	Iter		Err.
mean	1.0	0.05	0.001	0.397	0.603	0.000	22.6	525	1753	7E−06
S.E.	0.3	0.03	0.002	0.040	0.040	0.001	16.9	389	1513	1E−05
Min	0.1	0.00	0.000	0.340	0.511	0.000	5.4	168	389	2E−10
Max	1.5	0.10	0.009	0.489	0.660	0.004	70.1	1823	6851	4E−05

- Solving GMM is not as fast as solving MLE
  - the larger size of the moments problem
  - the nonlinearity introduced by the constraints related to moments, particularly the skewness equations.

Comparisons with NFXP

- We reduce time spent on solving DP
  - Important when DP is hard to solve
  - Less important as the cost of computing likelihood rises
- Closed-form solutions may hurt
  - Substituting out m variables from n squeezes all nonlinearities into the remaining $n - m$ variables that still appear in the objective and constraints.
  - Nonlinear elimination of variables reduces number of unknowns but may increase nonlinearity
  - Actually, it is often easier to solve large optimization problems!
  - In optimization, it is nonlinearity, not dimensionality, that makes a problem difficult.
- MPEC is far more flexible and easy to implementation.
  - Derivatives of both DP solution and likelihood are easier to compute
  - NFXP has a hard time doing analytic derivatives of DP step; uses finite differences
  - This approach encourages one to experiment with many solvers to find the best one

Comparisons with NFXP

- Ease of use
  - Rust used Gauss "because: 1. the GAUSS langauge is a high-level symbolic language which enables a nearly 1:1 translation of mathematical formulae into computer code. Matrix operations of GAUSS replace cumbersome do-loops of FORTRAN. 2. GAUSS has built-in linear algebra routines, no links to Lapack needed"
  - MPEC: AMPL is also easy to use. All solvers have access to linear algebra routines, many of which are better than Lapack. AMPL does not have matrix notation, but its approach to matrices, tensors, and indexed sets is very flexible.
- Optimization Method
  - Rust: Outer iteration uses BHHH for a while then switches to BFGS, where the user chooses the switch point.
  - JS: Use solvers far superior to these methods.

Comparisons with NFXP

- Derivatives
  - Rust: "The NFXP software computes the value of and its derivatives numerically in a subroutine. This implies that we can numerically compute and its derivatives for each trial value encountered in the course of the process of maximizing. In order to do this, we need a very efficient and accurate algorithm for computing the fixed point."
  - JS: Use true analytic derivatives. This is done automatically by AMPL, and is done efficiently using ideas from automatic differentiation.
- Dynamic programming method
  - Rust: "Inner Fixed Point Algorithm. Contraction mapping fixed point (poly)algorithm. The algorithm combines contraction iterations with Newton-Kantorovich iterations to efficiently compute the functional fixed point." In Rust, contraction iterations are linearly convergent; quadratic convergence is achieved only at final stage.
  - JS: We use Newton-style methods that are globally faster than contraction mapping ideas. This is particularly important if β is close to 1, representing short, but realistic, time periods.

Further Improvements

- Use continous methods to approximate value function and decision rules
  - True problem has a continuous state and a smooth value function
  - True solution can be computes accurately with far fewer free parameters than any discretization
- Estimate shock process instead of assuming a particular extreme value distribution
  - Use numerical integration methods to compute expectations
  - Estimate flexible functional form for distribution

Part IV

Estimation of Games

MPEC Approach to Games

- Suppose the game has parameters θ.
- Let σ denote the equilibrium strategy given θ; that is, σ is an equilibrium if and only if for some function G

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

- Suppose that likelihood of a data set, X, if parameters are θ and players follow strategy σ is $\mathcal{L}(\theta, \sigma, X)$. Therefore, maximum likelihood is the problem

$$\begin{aligned} \max_{(\theta,\sigma)} \quad & \mathcal{L}(\theta, \sigma, X) \\ \text{s.t.} \quad & 0 = G(\sigma, \theta) \end{aligned}$$

NFXP to Games

- NFXP requires finding all σ that solve $G(\theta, \sigma)$, compute the likelihood at each such σ, and report the max
- Finding all equilibria for arbitrary games is an essentially intractable problem - see Judd and Schmedders (2006) and mathematical literature
- In contrast, MPEC sends problem to good solvers. Multiple equilibria may produce multiple local solutions, but that is a standard problem in MLE, and would also be a problem for NFXP

Example: Pricing Game with Multiple Equilibria

- Bertrand game with 3 types of customers in 4 cities
  - Type 1 customers only want good x

$$Dx_1(p_{x,i}) = A - p_{x,i};\ Dy_1 = 0, \text{ for } i = 1, \ldots, 4.$$

  - Type 3 customers only want good y, and have a linear demand curve:

$$Dx_3 = 0;\ Dy_3(p_{y,i}) = A - p_{y,i}, \text{ for } i = 1, \ldots, 4.$$

  - Type 2 customers want some of both. Let n_i be the number of type 2 customers in a type i city.

$$\begin{aligned} Dx_2(p_{xi}, p_{yi}) &= n_i p_{xi}^{-\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma}\right)^{\frac{\gamma-\sigma}{-1+\sigma}} \\ Dy_2(p_{xi}, p_{yi}) &= n_i p_{yi}^{-\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma}\right)^{\frac{\gamma-\sigma}{-1+\sigma}} \end{aligned}$$

Example: Pricing Game with Multiple Equilibria

- Total demand for good x (y) in a type i city is given by

$$\begin{aligned} Dx(p_{xi}, p_{yi}) &= Dx_1(p_{xi}, p_{yi}) + Dx_2(p_{xi}, p_{yi}) \\ Dy(p_{xi}, p_{yi}) &= Dy_2(p_{xi}, p_{yi}) + Dy_3(p_{xi}, p_{yi}) \end{aligned}$$

- Let m be the unit cost of production for each firm. Revenue for good x (y) in a type i city is given by

$$\begin{aligned} Rx(p_{xi}, p_{yi}) &= (p_{xi} - m)Dx(p_{xi}, p_{yi}) \\ Ry(p_{xi}, p_{yi}) &= (p_{yi} - m)Dy(p_{xi}, p_{yi}) \end{aligned}$$

Example: Pricing Game with Multiple Equilibria

- Let MR_x be marginal profits for good x; similarly for MR_y.

$$MR_x(p_{xi}, p_{yi}) = A - p_{xi} + n_i \left(p_{xi}^{\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\gamma-\sigma}{\sigma-1}} \right)^{-1}$$

$$+ (p_{xi} - m) \left(-1 + \frac{n_i(\sigma - \gamma)}{p_{xi}^{2\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{1+\frac{\sigma-\gamma}{\sigma-1}}} - \frac{n_i \sigma}{p_{xi}^{1+\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\sigma-\gamma}{\sigma-1}}} \right)$$

$$MR_y(p_{xi}, p_{yi}) = A - p_{yi} + n_i \left(p_{yi}^{\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\gamma-\sigma}{\sigma-1}} \right)^{-1}$$

$$+ (p_{yi} - m) \left(-1 + \frac{n_i(\sigma - \gamma)}{p_{yi}^{2\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{1+\frac{\sigma-\gamma}{\sigma-1}}} - \frac{n_i \sigma}{p_{yi}^{1+\sigma} \left(p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\sigma-\gamma}{\sigma-1}}} \right)$$

Example: Pricing Game with Multiple Equilibria

- Assume that the four markets differ only in terms of type 2 customer population, n_i; we assume they are $(n_1, n_2, n_3, n_4) = (1500, 2500, 3000, 4000)$
- The other parameters are common across markets:

$$\sigma = 3;\ \gamma = 2;\ m = 1;\ A = 50$$

- For each city, we solve the FOC

$$\left.\begin{array}{l} MR_x(p_{xi}, p_{yi}) = 0 \\ MR_x(p_{xi}, p_{yi}) = 0 \end{array}\right\} \text{ for } i = 1, \ldots, 4$$

and check the second-order conditions global optimality for each firm in each potential equilibria

Example: Pricing Game with Multiple Equilibria

- Equilibrium possibilities for each firm
  - Niche strategy: price high, get low elasticity buyers.
  - Mass market strategy: price low to get type 2 people.
  - Low population implies both do niche
  - Medium population implies one does niche, other does mass market, but both combinations are equilibria.
  - High population implies both go for mass market

Example: Pricing Game with Multiple Equilibria

- Unique equilibrium for type 1 and 4 cities:

$$\begin{array}{lrcl} \text{city 1:} & (p_{x1}, p_{y1}) & = & (24.24, 24.24) \\ \text{city 4:} & (p_{x4}, p_{y4}) & = & (1.71, 1.71) \end{array}$$

  - In type 1 cities is for both firms to choose a niche strategy
  - In type 4 cities is for both to pursue a mass market strategy
  - Intuition: the mass market is small in type 1 cities but large in type 4 cities.

Example: Pricing Game with Multiple Equilibria

- There are two equilibria in type 2 cities and type 3 cities:

$$\begin{array}{lll} \text{City 2:} & \left(p^I_{x2}, p^I_{y2}\right) & = \quad (25.18,\ 2.19) \\ & \left(p^{II}_{x2}, p^{II}_{y2}\right) & = \quad (2.19,\ 25.18) \\ \\ \text{City 3:} & \left(p^I_{x3}, p^I_{y3}\right) & = \quad (2.15,\ 25.12) \\ & \left(p^{II}_{x3}, p^{II}_{y3}\right) & = \quad (25.12,\ 2.15) \end{array}$$

- One firm chooses a niche strategy while the other chooses a mass market strategy.
- Intuition: the mass markets in type 2 and type 3 cities are large enough to attract one firm, but not large enough to attract the other firm.

Generating Synthetic Data

- Assume that the equilibria in the four city types are

$$\begin{aligned}(p_{x1}, p_{y1}) &= (24.24, 24.24)\\ (p_{x2}, p_{y2}) &= (25.18,\ 2.19)\\ (p_{x3}, p_{y3}) &= (2.15,\ 25.12)\\ (p_{x4}, p_{y4}) &= (1.71, 1.71)\end{aligned}$$

- Econometrician observes price data with measurement errors for 4K cities, with K cities of each type
- We used a normally distributed measurement error $\varepsilon \sim N(0, 50)$ to simulate price data for 40,000 cities, with 10,000 cities of each type ($K = 10{,}000$)
- We want to estimate the unknown structural parameters (σ, γ, A, m) as well as equilibrium prices $(p_{xi}, p_{yi})_{i=1}^{4}$ implied by the data in all four cities.

Example: Pricing Game with Multiple Equilibria

- MPEC formulation

$$\begin{aligned}
\min_{(p_{xi},p_{yi},\sigma,\gamma,A,m)} \quad & \sum_{k=1}^{K}\sum_{i=1}^{4}\left((p_{xi}^k - p_{xi})^2 + (p_{yi}^k - p_{yi})^2\right) \\
\text{subject to:} \quad & p_{xi}, p_{yi} \geq 0, \ \forall i \\
\text{[FOC:]} \quad & 0 = MR_y(p_{xi}, p_{yi}) = MR_x(p_{xi}, p_{yi}), \ \forall i \\
\text{[global opt:]} \quad & (p_{xi} - m)Dx(p_{xi}, p_{yi}) \geq (p_j - m)Dx(p_j, p_{yi}), \ \forall i, j \\
\text{[global opt:]} \quad & (p_{yi} - m)Dy(p_{xi}, p_{yi}) \geq (p_j - m)Dy(p_{xi}, p_j), \ \forall i, j
\end{aligned}$$

- We do not impose an equilibrium selection criterion

Game Estimation Results

- Case 1: Estimate only σ and γ and fix $A_x = A_y = 50$ and $m_x = m_y = 1$
- Case 2: Estimate all six structural parameters but impose the symmetry constraints on the two firms: $A_x = A_y$ and $m_x = m_y$
- Case 3: Estimated all six structural parameters without imposing the symmetry constraints

Parameters	Case 1	Case 2	Case 3
(σ, γ)	$(\ 3.008,\ \ 2.016)$	$(\ 2.822,\ \ 1.987)$	$(\ 3.080,\ \ 2.093)$
(A_x, A_y)		$(50.404, 50.404)$	$(50.236, 49.544)$
(m_x, m_y)		$(\ 0.975,\ \ 0.975)$	$(\ 1.084,\ \ 0.972)$
(p_{x1}, p_{y1})	$(24.292, 24.292)$	$(24.443, 24.443)$	$(24.694, 24.241)$
(p_{x2}, p_{y2})	$(25.192,\ \ 2.166)$	$(25.248,\ \ 2.139)$	$(25.434,\ \ 2.004)$
(p_{x3}, p_{y3})	$(\ 2.127, 25.135)$	$(\ 2.100, 25.163)$	$(\ 2.243, 24.934)$
(p_{x4}, p_{y4})	$(\ 1.718,\ \ 1.718)$	$(\ 1.730,\ \ 1.730)$	$(\ 1.814,\ \ 1.652)$

Conclusion

- Structural estimation methods are far easier to construct if one uses the structural equations
- The numerical algorithm advances of the past forty years (SQP, Augmented Lagragian, Interior Point, AD, MPEC) with NLP solvers such as KNITRO, SNOPT, filterSQP, PATH, makes this tractable
- Numerical analysis is more useful for empirical economists than new econometric theory
- User-friendly interfaces (e.g., AMPL, GAMS) makes this as easy to do as Stata, Gauss, and Matlab
- This approach makes structural estimation *really* accessible to a larger set of researchers