A New Optimization Approach to Maximum Likelihood Estimation of Structural Models

Che-Lin Su
Northwestern University

Kenneth L. Judd
Hoover Institution

Originally presented in November, 2006 at Stanford GSB
This verison: March, 2021

Structural Estimation

Great interest in estimating models based on economic structure

- Dynamic programming models
- Games
- Dynamic stochastic general equilibrium

Major computational challenge because estimation involves also solving the model

We show that many computational difficulties can be avoided by using state-of-the-art optimization algorithms and software

Basic Problem - DP Example

- Individual solves a dynamic programming problem
  - Problem is parameterized by unobserved parameters θ
  - Solution is the set of decision rules σ
  - Solution also includes the value function which is generally eliminated by some algebra reduction, so we will ignore that here
- The data X: Econometrician observes state (with error) and decisions
- Defines (somehow) the likelihood function for data X

$$L(\theta; X)$$

- Rationality imposes a relationship between θ and σ

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

- We want to find maximum likelihood θ but impose rationality condition
- You would like to find a function $\sigma = \Sigma(\theta)$ such that

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

  telling you what σ is for a θ.

  - Problem 1: This is difficult, if not impossible
  - Problem 2: There may be multiple σ for some θ

The augmented likelihood function

- Definition: The augmented likelihood function is the likelihood of X given θ, the set of parameters, and σ, the decision rule, and is denoted
$$\mathcal{L}\left(\theta,\sigma;X\right)$$
  - Well-defined because the likelihood is a function of theta and sigma separately
  - Creates greater flexibility in thinking about computational methods
- In general, increasing the dimensionality of a problem allows us to examine each component in isolation
- Likelihood function is defined by
$$L\left(\theta;X\right)=\mathcal{L}\left(\theta,\Sigma\left(\theta\right);X\right)$$

Nested Fixed Point Algorithm

- Find theta that solves
$$\max L(\theta; X) = \mathcal{L}(\theta, \Sigma(\theta); X)$$
  - Inner loop: takes a theta and computes $\Sigma(\theta)$
  - Outer loop: iterate over θ guesses to find best
- Problem: Must compute $\sigma = \Sigma(\theta)$ to high accuracy for each θ guess
- 2004 View: Erdem et al. (2004):

  > 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). ....[S]ome recent research ... proposes COMPUTATIONALLY SIMPLE estimators for structural models including auctions, demand in differentiated product markets, dynamic discrete choice and dynamic games. 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.

- Su-Judd argues against computationally simple or light methods, and produced full information maximum likelihood estimates

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 (from Econ 712, University of Wisconsin, 1979)
  - Assume, for example, a functional form for utility

$$u\left(c\right) = c - \beta c^2.$$

  - Solve for demand function

$$c = \left(1 - p\right) / \left(2\beta\right)$$

  - Hence, i'th data point satisfies

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

    for some ε_i.
  - To estimate β, choose β to minimize the sum of squared errors

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

- Limitations
  - Need to solve for demand function, which is hard if not impossible
  - For example, suppose
    $$u(c) = c - \beta\left(c^2 + c^4 + c^6\right)$$
    with first-order condition
    $$1 - \beta\left(2c + 4c^3 + 6c^5\right) = p$$
  - There is no closed-form solution for demand function.
  - What were you taught to do in this case? *Change the model*!
- MPEC 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 but are consistent with structure

- Examples
  - For our consumption demand model, this is the problem

$$\begin{aligned} \min_{\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & u_c\left(q_i - \varepsilon_i; \beta\right) = p_i \end{aligned}$$

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

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & 1 - 2\beta c_i = p_i \\ & q_i = c_i + \varepsilon_i \end{aligned}$$

  - Degree-six utility function produces problem

$$\begin{aligned} \min_{c_i,\varepsilon_i,\beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & 1 - \beta\left(2c_i + 4c_i^3 + 6c_i^5\right) = p_i \\ & q_i = c_i + \varepsilon_i \end{aligned}$$

- Don't get confused over the appearance of the epsilons as "estimated" variables
  - Normal least-squares also "estimates" the epsilons
  - BOTH LS and MPEC produce the same estimates
  - Reply to Berry and Conlon: If MPEC has {nuisance, incidental} {variable, parameter} problems than so does LS

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

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

    * 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.
- Using closed-form solution is often a bad idea
  - Substituting out m variables from n squeezes all nonlinearities into the remaining $n-m$ dimensions.
  - Nonlinear elimination of variables reduces number of unknowns but may increase nonlinearity
  - Actually, it is often easier to solve large optimization problems!
    - Nonlinear elimination of variables reduces number of unknowns but may increase nonlinearity
    - Bigger system of equations may be very sparse; number of nonzero elements of Jacobian may be much less
  - In optimization, it is nonlinearity, not dimensionality, that makes a problem difficult.
- Implicit formulations of problems may add variables but are more flexible and seldom adds unacceptable computational burdens.

Basic Problem - DP Example

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

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

  where θ is set of parameters and σ is decision rule

- Rationality imposes a relationship between θ and σ

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

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

MPEC Approach

- Maximum likelihood is the constrained optimization problem

$$\begin{array}{ll} \max\limits_{\sigma,\theta} & \mathcal{L}\left(\theta,\sigma;X\right) \\ \text{s.t.} & 0 = G\left(\theta,\sigma\right) \end{array}$$

  - Solution does produce $\sigma = \Sigma\left(\theta\right)$ for θ solution
  - Does not compute $\sigma = \Sigma\left(\theta\right)$ for each θ
  - If problem is smooth, can use quadratically convergent methods for constrained optimization

- Formulating a problem as a constrained optimization problem allows one to use many alternative methods from numerical analysis

- Lesson: Define a problem before describing a computational method to solve it

MPEC Applied to Zurcher (Su-Judd, 2012)

- MPEC is faster than NFXP
- Timing is nearly linear in the number of states for modest grid size.
- The likelihood function, the constraints, and their derivatives are evaluated only 45-182 times in this example.
- In contrast, the Bellman operator in NFXP (the constraints here) is evaluated hundreds (thousands?) of times in NFXP.
- Rust did a two-stage procedure, estimating transition parameters in first stage. We do FIML

Inference and Parametric Bootstrap

- One could compute standard errors with the formulas in Aitchison-Silvey (1958)
- Bootstrapping can be used to generate standard errors and are likely better
- We did 20 resamplings:
  - Five parameter estimation
  - 1000 data points
  - 1001 grid points in DP

Comparison of Su-Judd MPEC implementation with Rust Implementation

- Ease of use
  - Rust used Gauss because:
    - GAUSS 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.
    - GAUSS has built-in linear algebra routines, no links to Lapack needed"
  - MPEC:
    - AMPL (GAMS, AIMMS,...) is also easy to use.
    - All solvers have access to GOOD linear algebra routines.
    - AMPL does not have matrix notation, but its approach to 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.
  - MPEC: Use solvers far superior to these methods.

- 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."
  - MPEC:
    - Automatically computes efficient code for the true analytic derivatives.
    - It also automatically detects sparseness properties of the Jacobian of the constraints
- 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.
  - MPEC: We use quadratically convergent Newton-style methods for the whole problem.

1986 versus 2006 versus 2021

- RAM: explosion in capacity
  - 64K was good in 1986. 64M was small in 2006.
  - NFXP is an example of a low-memory algorithm
- Solvers: tremendous improvements
  - 1986: MINOS was available, but likely limited by RAM
  - 2006: NPSOL, SNOPT, KNITRO, Filter, Conopt, ...
  - 2021: IPOPT is free. fmincon is neither free nor reliable

- Derivatives:
  - 1986: Hand-coded or finite-difference
  - 2006: Automatic differentiation (AD) widely available
  - 2021: AD for Matlab possible only with add-ons (which are free and buggy)
- Preconditioning
  - 1986: By hand
  - 2006: Automatic in MINO, NPSOL, SNOPT and others
  - 2021: By hand in Matlab
- Automatic sparseness detection
  - 1986: A fantasy
  - 2006: Implemented in AMPL and GAMS
  - 2021: Not available in Matlab

- Modeling languages
  - 1986: GAMS for general equilibrium models
  - 1992: DICE uses GAMS/Conopt
  - 2006: AMPL and GAMS widely used for all kinds of models
  - 2021: Now have AIMMS and others being developed; Matlab?
- Computing power
  - 1986 to 2006: Moore's law implies 1000-fold increase.
  - 2008: Yongang Cai's thesis solves DP problems in parallel at UW using ˜200 workstations with HTCondor
  - 2013: Cai-Judd-Lontzek uses 80,000 cores to solve a DSICE case (including 300 billion optimization problems) in about five hours (linear scaling)
  - 2014: Yeltekin-Cai-Judd uses 160,000 cores to solve supergames (linear scaling)
  - 2021; Judd-Mueller solves AMSS model (40,000 state discrete-state DP) in about a minute, using a 3500-core GPU, paying $10/month

- Cost of computing power
  - 1986: Pay for desktop and Gauss
  - 2006: Use of AMPL and GAMS is free at NEOS; HTCondor is free
  - 2012: Cai and Judd gets first of five Blue Waters allocations (~20M core hours)
  - 2019: Rangel parallelizes NEOS
  - 2021: Matlab is expensive and limited in parallelizability
- Conclusion
  - 1986: NFXP was the only way to do structural estimation
  - 2006: 20 years produced advances in hardware and software that made far superior methods available
  - 2021: Another 15 years, explosion in computational capability, and plummeting costs
- Question:
  - Do you listen to the same music your parents did?
  - Do you wear the same clothes your parents wore?
  - Do you use the same cellphone your parents used?

2021 versus the future

- Massive parallelism
  - NFXP could compute likelihood function at many theta values simultaneously across thousands of cores.
  - Constrained optimization algorithms are written in a serial manner
    - Some pieces could be parallelized – matrix-vector operations, evaluating separable functions
    - Load balancing would be complicated
    - Math programming people are pessimistic
- Possible NFXP inner loop improvements
  - Use nonlinear equation solver
  - Exploit sparseness, analytic derivatives

- Inference
  - Good confidence intervals require MPEC formulations
  - See Reich-Judd next week
- Bayesian methods
  - Exascale (grid, highpower, highthrouput, BOINC, ...) computing make Bayesian methods feasible.
  - Finding local optima of likelihood function is a small part of computational effort. NFXP vs. MPEC becomes irrelevant
  - Choice of quadrature rule would be the critical factor
  - Any volunteers for doing Bayesian Zurcher bus problem? Bayesian BLP?

The MPEC vs NFXP Approach to games

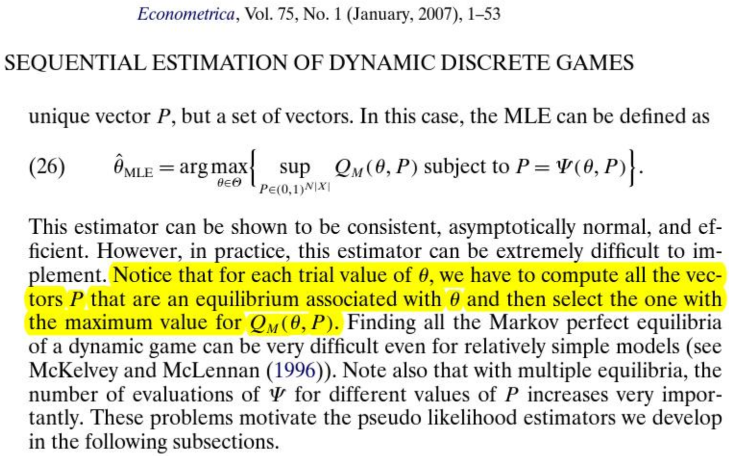
- Suppose that the game has parameters θ representing payoffs, probabilities, and whatever else is not observed directly by the econometrician.
- For a given function G, σ is an equilibrium given θ if and only if

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

- Suppose that augmented likelihood of a data set, X, is $\mathcal{L}(\theta, \sigma, X)$. Therefore, maximum likelihood is the problem

$$\begin{array}{ll} \max\limits_{\sigma,\theta} & \mathcal{L}(\theta, \sigma, X) \\ \text{s.t.} & 0 = G(\sigma, \theta) \end{array}$$

CW: Estimation of Games: January, 2007



Econometrica, Vol. 75, No. 1 (January, 2007), 1–53

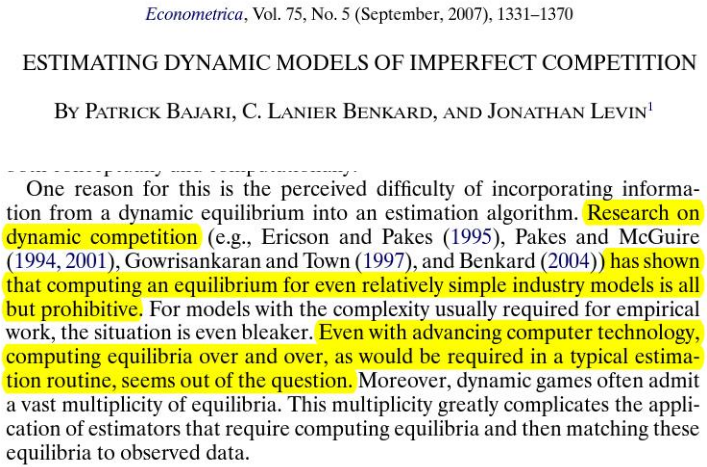
SEQUENTIAL ESTIMATION OF DYNAMIC DISCRETE GAMES

unique vector P, but a set of vectors. In this case, the MLE can be defined as

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta\in\Theta}\left\{ \sup_{P\in(0,1)^{N|X|}} Q_M(\theta, P) \text{ subject to } P = \Psi(\theta, P)\right\}. \tag{26}$$

This estimator can be shown to be consistent, asymptotically normal, and efficient. However, in practice, this estimator can be extremely difficult to implement. Notice that for each trial value of θ, we have to compute all the vectors P that are an equilibrium associated with θ and then select the one with the maximum value for $Q_M(\theta, P)$. Finding all the Markov perfect equilibria of a dynamic game can be very difficult even for relatively simple models (see McKelvey and McLennan (1996)). Note also that with multiple equilibria, the number of evaluations of Ψ for different values of P increases very importantly. These problems motivate the pseudo likelihood estimators we develop in the following subsections.

CW: Estimation of Games: September, 2007



Econometrica, Vol. 75, No. 5 (September, 2007), 1331–1370

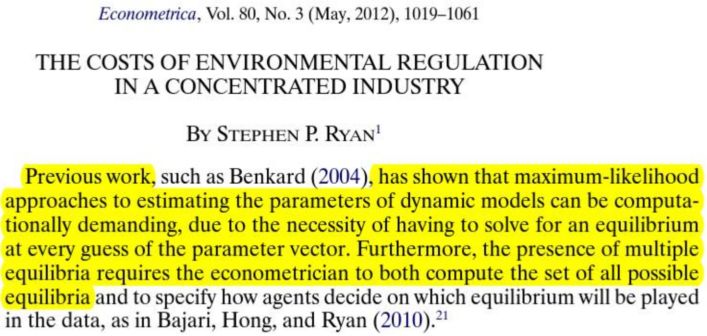
ESTIMATING DYNAMIC MODELS OF IMPERFECT COMPETITION

BY PATRICK BAJARI, C. LANIER BENKARD, AND JONATHAN LEVIN[1]

[illegible]

One reason for this is the perceived difficulty of incorporating information from a dynamic equilibrium into an estimation algorithm. Research on dynamic competition (e.g., Ericson and Pakes (1995), Pakes and McGuire (1994, 2001), Gowrisankaran and Town (1997), and Benkard (2004)) has shown that computing an equilibrium for even relatively simple industry models is all but prohibitive. For models with the complexity usually required for empirical work, the situation is even bleaker. Even with advancing computer technology, computing equilibria over and over, as would be required in a typical estimation routine, seems out of the question. Moreover, dynamic games often admit a vast multiplicity of equilibria. This multiplicity greatly complicates the application of estimators that require computing equilibria and then matching these equilibria to observed data.

CW: Estimation of Games: 2012



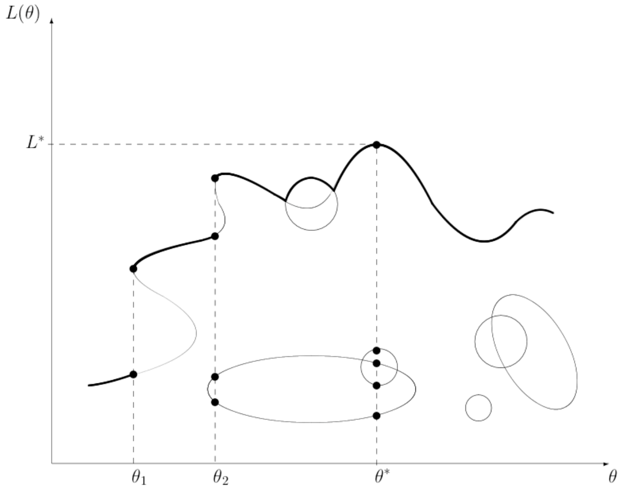
Econometrica, Vol. 80, No. 3 (May, 2012), 1019–1061

THE COSTS OF ENVIRONMENTAL REGULATION
IN A CONCENTRATED INDUSTRY

BY STEPHEN P. RYAN[1]

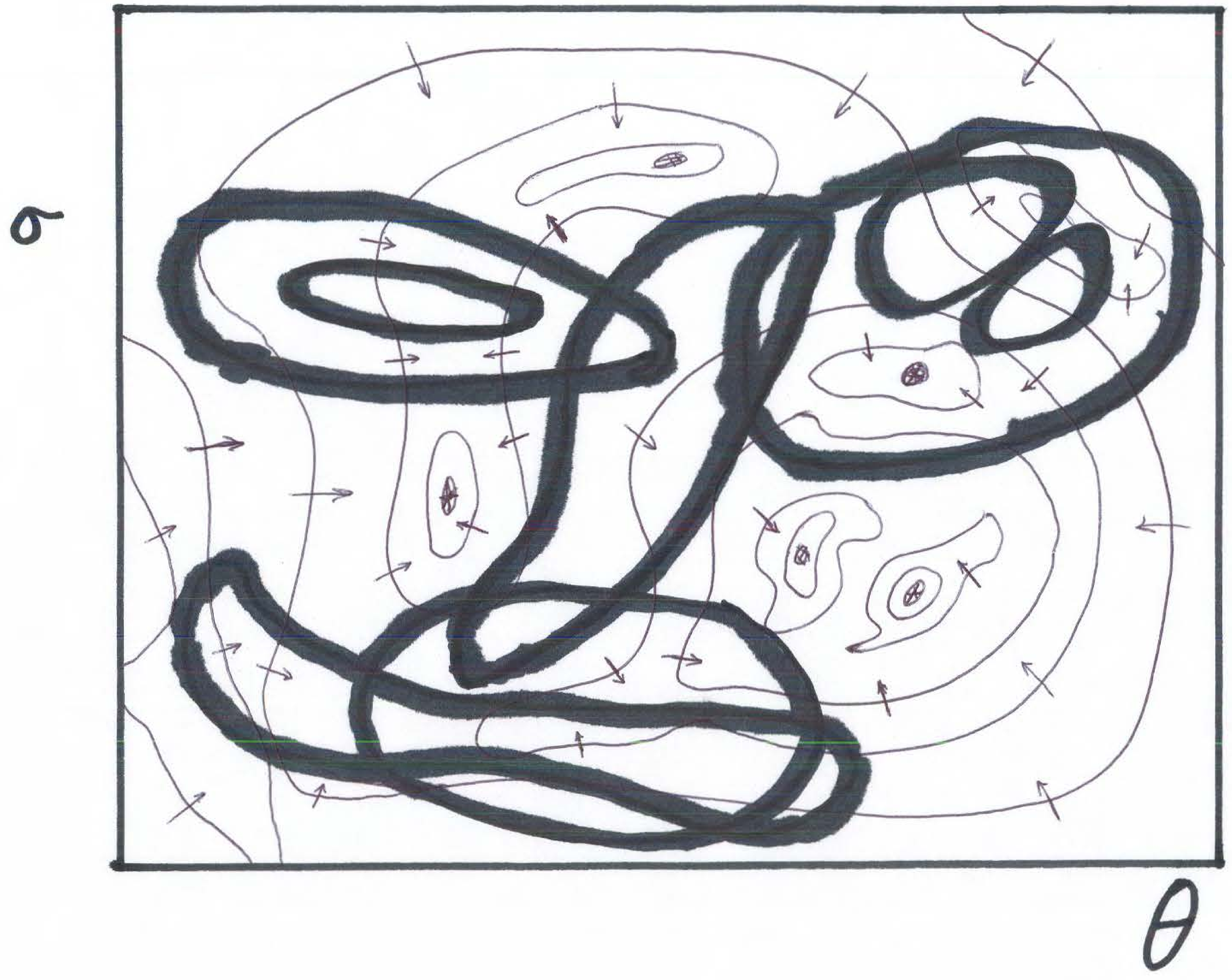
Previous work, such as Benkard (2004), has shown that maximum-likelihood approaches to estimating the parameters of dynamic models can be computationally demanding, due to the necessity of having to solve for an equilibrium at every guess of the parameter vector. Furthermore, the presence of multiple equilibria requires the econometrician to both compute the set of all possible equilibria and to specify how agents decide on which equilibrium will be played in the data, as in Bajari, Hong, and Ryan (2010).[21]

NFXP Applied to Games with Multiple Equilibria



- For each θ, NFXP would require the inner loop to
  - find all the σ that solves $G(\sigma, \theta)$, an intractable problem
  - compute likelihood at each equilibrium σ, and
  - report the max.
- In contrast, MPEC just sends the problem to good optimization solvers.
- Multiplicity of local max is a property of the likelihood function, not the algorithm
- Multiple local solutions is a common problem in estimation

MPEC



CONSTRAINED OPTIMIZATION APPROACHES TO ESTIMATION OF STRUCTURAL MODELS: COMMENT[*]

FEDOR ISKHAKOV
UNIVERSITY OF NEW SOUTH WALES

JOHN RUST[†]
GEORGETOWN UNIVERSITY

BERTEL SCHJERNING
UNIVERSITY OF COPENHAGEN

JUNE 2014

Abstract

We revisit the comparison of mathematical programming with equilibrium constraints (MPEC) and nested fixed point (NFXP) algorithms for estimating structural dynamic models by Judd and Su (JS, 2012). They used an inefficient version, NFXP-SA, that relies on the method of successive approximations to solve the fixed point problem. We re-do their comparison using the more efficient version of NFXP that Rust (1987) used, NFXP-NK, which combines successive approximations and Newton-Kantorovich iterations to solve the fixed point problem. MPEC and NFXP-NK are similar in performance when the fixed point dimension and sample size are relatively small and the discount factor is not too close to 1. However for higher dimensional problems, or problems with large sample sizes, NFXP-NK outperforms MPEC by orders of magnitude. MPEC fails to converge with high probability as the fixed point dimension increases, or as the discount factor approaches 1.

Rust et al. Critiques

- June, 2014, criticism
  - NFXP outperforms MPEC for problems of high dimension
  - MPEC often fails to converge
- Che-Lin Su response later in 2014
  - Rust et al. did not use the same equations to describe the DP problem
    - They used an ill-conditioned version for MPEC
    - They used a preconditioned version for NFXP
  - When the same equations were used, NFXP and MPEC performance was basically the same
    - Both succeeded well with the preconditioned version
    - Both failed with the ill-conditioned version

- Published version (after shotgun merger with Lee and Seo)
  - Claimed NFXP was often faster than MPEC
  - Used only the Rust bus example for comparison
    - Did not look at BLP examples of Dube-Fox-Su
    - Did not look at game examples in Su-Judd, 2011.
- Judd response in 2021: Definition of "time" cost was skewed in favor of NFXP
  - Both used analytic derivatives and sparse representations of the Jacobian
  - Rust et al. only includes running time of code; ignores the many hours (days?) it takes to compute and code up analytic derivates and sparseness
  - AMPL plus solver does everything – efficient analytic derivatives, construction of sparse Jacobian, preconditioning and execution – in a short amount of time.

MPEC Approach to Method of Moments

- Suppose you want to fit moments. E.g., likelihood may not exist in most of parameter space.
- Method then is

$$\begin{array}{ll} \min_{\sigma,\theta} & \left\| M^{\text{ model }}(\sigma,\theta) - M^{\text{ data }}(X) \right\|^2 \\ \text{s.t.} & 0 = G(\sigma,\theta) \end{array}$$

  - Could compute $M^{\text{ model }}(\sigma,\theta)$ numerically via integral equation methods
  - Could simulate moments in $M^{\text{ model }}(\sigma,\theta)$

MPEC Approach to GMM

- Efficient GMM estimation requires the use of the proper variance-covariance matrix W, which is a function of the estimates.
- Standard method iterates on W until it satisfies $W = \mathcal{W}(X)$ for some variance-covariance function $\mathcal{W}(X)$
- MPEC method is

$$\begin{aligned} \min_{\sigma,\theta,W} \quad & GMM\left(\sigma,\theta,W,X\right) \\ \text{s.t.} \quad & 0 = G\left(\sigma,\theta,W,X\right) \\ & 0 = W - \mathcal{W}(X) \end{aligned}$$

Conclusion

- Structural estimation methods are far easier to construct if one includes the structural equations.
- The numerical algorithm advances of the past forty years (SQP, augmented Lagrangian, interior point, AD, MPCC) makes this tractable
- User-friendly interfaces (e.g., AMPL) makes this as easy to do as Stata, Gauss, and Matlab
- This approach makes structural estimation *really* accessible to a larger set of researchers.