# Constrained Optimization Approaches to Estimation of Structural Models

Kenneth L. Judd
Hoover Institution

Che-Lin Su
Northwestern University

February 2006
This version: July, 2007*

Abstract. Maximum likelihood estimation of structural models is regarded as computationally difficult. This impression is due to a focus on the Nested Fixed-Point approach. We present a direct optimization approach to the problem and show that it is significantly faster than the NFXP approach when applied to the canonical Zurcher bus repair model. The NFXP approach is inappropriate for estimating games since it requires finding all Nash equilibria of a game for each parameter vector considered, a generally intractable computational problem. We reformulate the problem of maximum likelihood estimation of games into an optimization problem that is qualitatively no more difficult to solve than standard problems. The direct optimization approach is also applicable to other structural estimation problems such as auctions and RBC models, and also to other estimation strategies, such as the methods of moments. It is also easily implemented on standard software.

## 1. Introduction

Structural estimation of economic models is an important technique for analyzing economic data. However, it is commonly believed that computational demands make it difficult to implement the most powerful statistical methods. For example, maximum likelihood estimation is the gold standard in estimation and Rust (1987) provided a strategy for maximum likelihood estimation of economic models, an approach he dubbed the Nested Fixed-Point (NFXP) algorithm. Unfortunately, NFXP is computationally demanding since it repeatedly takes a guess for the structural parameters and solves for the corresponding endogenous economic variables with high accuracy. While it is clear that NFXP is impractical in many contexts, particularly in the estimation of games, this does not mean that maximum likelihood estimation is intractable. We present a direct optimization approach, called the MPEC (Mathematical Programming with Equilibrium Constraints) to structural estimation that avoids repetitive solution of the structural model; in fact, the only equilibrium

*This version was prepared for the 2007 Chicago-Argonne Institute of Computational Economics, July 30-August 9. This version is incomplete. We welcome comments.

that is solved is the one associated with the final estimate. The idea behind the MPEC[1] approach is simple: choose structural parameters and endogenous economic variables so as to maximize likelihood subject to the constraint that the endogenous economic variables are consistent with an equilibrium consistent with the structural parameters. That's it. Nothing more. When formulated in this way, it is clear that all we do is write down expressions defining the likelihood (the objective) and the equilibrium equations (the constraints), and submit them to one or more of the state-of-the-art constrained optimization programs. These solvers treat the parameters and endogenous variables jointly and do not repeatedly solve for equilibria, they will be faster than any method that imposes equilibrium values for every guess of the structural parameters. Furthermore, since this approach allows us to directly use state-of-the-art algorithms, an economist avoids all decisions about algorithmic details (such as, for example, choosing between BHHH and BFGS, choosing between value function iteration or policy iteration, etc.) that makes NFXP and related methods costly for a user to implement even when a model is tractable.

Of course, NFXP and other procedures are solving the same constrained optimization problem. However, NFXP proceeds in a nested manner. While nesting may sound intuitive to economists, it is not computationally efficient. NFXP is really numerical nonlinear elimination of variables, an approach never taken in numerical optimization. Instead, numerical algorithms for constrained optimization usually proceed by introducing shadow prices for the constraints, writing down the Lagrangian, and analyzing the resulting objective with a combination of various ideas (Newton's method, linesearch, trust region, sequential quadratic approximation, etc.). The general observation is that nesting is a waste of time because one spends a lot of effort solving the constraints (e.g., implementing nonlinear elimination of variables) at points far from the solution. All the leading optimization methods avoid this waste.

We first illustrate the idea in a simple demand example, related to the constrained maximum likelihood literature. We then show that the MPEC approach is significantly faster than the NFXP approach when applied to the canonical Zurcher bus repair model even though we use no special feature of the problem. To demonstrate this we also implement a method of moments estimator to the bus problem.

Our final examples show that the MPEC approach is immediately applicable data from games, even when structural parameters do not imply a unique equilibrium. The NFXP approach is infeasible[2] for estimating all but the simplest games since it requires finding

[1]This approach often fits into a variety of mathematical categories, such as bilevel optimization, inverse optimal problems, MPEC, EPEC, or MPCC problems. We have chosen to use the term MPEC because it is a good fit for the ideas in this paper and in future generalizations.

[2]In the interest of precision, by "infeasible" we mean using the state of the art mathematical algorithms

all Nash equilibria of a game for each parameter vector considered. The game theory and numerical analysis literatures clearly show that this is an intractable computational problem for all but a few, special, and small games[3]. We reformulate the problem of maximum likelihood estimation of games as an optimization problem that is qualitatively no more difficult to solve than standard maximum likelihood problems. Not only will this approach produce the maximum likelihood estimate, but it avoids the difficulties of other methods which are less efficient statistically and often need to make additional structural assumptions.

On a more general note, this paper builds on ideas and methods for that were developed and analyzed in the statistics and econometrics literatures. In particular, all of the examples below are constrained estimation problems of the kind exposited in Aitchison and Silvey (1958), Silvey (1970), Gallant and Holly (1980), Wolak (1987), Gallant and Tauchen (1989), Wolak (1989).

## 2. Current Views of Econometricians

The econometrics literature contains much pessimism regarding full maximum likelihood estimation of structural models. For example, Erdem et al. (2004) asserts the following in a section titled "Reducing the Computational Burden of Structural Estimation":

> 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 including auctions, demand in differentiated product markets, dynamic discrete choice and dynamic games. The estimators ... use a two-step approach. In the first step, one flexibly estimates a reduced form for agents' behavior consistent with the underlying structural model. In the second step, the one recovers the structural parameters, by plugging the first-step estimates into the model.....The two-step estimators can have drawbacks. First, there can be a loss of efficiency. The parameters estimated in the second step will depend on a nonparametric first step. If this first step is imprecise, the second step will be poorly estimated. Second, stronger

and supercomputers running at the top known 2007 speed of 200 TFlops, such an approach would take at least many days to solve the problem.

[3]See Judd and Schmedders (2005) for a case where finding all solutions is feasible and a general discussion of the relevant mathematics.

assumptions about unobserved state variables may be required. In a dynamic discrete choice model, accounting for unobserved heterogeneity by using random effects or even a serially correlated, unobserved state variable may be possible using a nested fixed point approach. 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.

The prevailing pessimism about structural estimation is particularly strong when it comes to the discussion of games. For example, Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) state that

> [M]ost applications of empirical discrete games have estimated static models. Two main econometric issues have limited the scope of applications to relatively simple static games: the computational burden in the solution of dynamic discrete games, and the indeterminacy problem associated with the existence of multiple equilibria.....
>
> The existence of multiple equilibria is a prevalent feature in most empirical games where best response functions are non-linear in other players' actions. Models with multiple equilibria do not have a unique reduced form and this incompleteness may pose practical and theoretical problems in the estimation of structural parameters. In particular, maximum likelihood and other extremum estimators require that we obtain all the equilibria for every trial value of the parameters. This can be infeasible even for simple models.

They make this more precise with the following discussion:

> Define the following pseudo likelihood function:
>
> $$Q_M(\theta, P) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{N} \ln \Psi_i(a_{imt}|x_{mt}; P, \theta) \tag{24}$$
>
> where $P$ is an arbitrary vector of players' choice probabilities. We call this function a pseudo likelihood because the choice probabilities are not necessarily equilibrium probabilities associated with $\theta$, but just best responses to an arbitrary vector $P$. Consider first the hypothetical case of a model with a unique equilibrium for each possible value of $\theta \in \Theta$. Then, the maximum likelihood estimator (MLE) of $\theta^0$ can be defined from the following constrained multinomial likelihood:
>
> $$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} Q_M(\theta, P) \text{ subject to: } P = \Psi(\theta, P) \tag{25}$$

> The computation of this estimator requires one to evaluate the mapping $\Psi$ and the Jacobian matrix $\delta\Psi/\delta P'$ at many different values of $P$. Though evaluations of $\Psi$ for different $\theta'$s can be relatively cheap because we do not have to invert the matrix $(I - \beta F)$ in (14), evaluations for different $P$ imply a huge cost when the dimension of the state space is large because this matrix needs to be inverted each time. Therefore, this estimator can be impractical if the dimension of $P$ is relatively large. For instance, that is the case in most models with heterogenous players because the dimension of the state space increases exponentially with the number of players. For that type of models this estimator can be impractical even when the number of players is not too large.

Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) are correct in stating that the MLE problem is a constrained optimization problem. However, their characterization of the literature on numerical methods for constrained optimization problems is out of date by over forty years. Even the early work by Fiacco and McCormick avoided any numerical elimination of variables. It is not now true that "evaluations for different $P$ imply a huge cost when the dimension of the state space is large because this matrix needs to be inverted each time." In fact, linear equation solvers never invert a matrix, instead computing an LU decomposition and then using backsolving to compute the solution. More commonly, update methods replacing the LU decomposition with updates on the most recent guess.

Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) also appear to worry about the cost of computing $\delta\Psi/\delta P'$, contradicting the applied math literature literature on computing Jacobians. Suppose that $f : R^n \to R^m$, and that the cost of computing $f$ is $t$ units of time. Then, the cost of computing the Jacobian of $f$, $f^i_j$, is $5t$ units of time independent of the number of variables $n$. The fact that computing a Jacobian is not expensive has been known since the 1960's. Leigh Tesfatsion's work is an example of an economist contributing to this literature. Of course, Aguirregabiria and Mira (Econometrica, 2007, pp. 1-53) are correct in thinking that computing Jacobians and inverting matrices *could* be impediments to a constrained optimization approach, as would be the case if one were using finite differences and not exploiting any sparseness. But these considerations are not important today; in fact, sparse matrix methods were include in early releases of MINOS over 20 years ago. Their assertion that a constrained optimization approach "can be impractical if the dimension is relatively large" is vacuously true since any numerical method becomes impractical if the dimension becomes large such as would be the case if $P$ had $10^9$ dimensions. However, this observation is relevant only if the constrained optimization approach is impractical at sizes for which their alternative is practical. They chose to offer no evidence on that point.

Aguirregabiria and Mira (Econometrica, 2007, pp. 1-53) continue their discussion of constrained optimization problems:

> An important complication in the estimation of dynamic games is that for some values of the structural parameters the model can have multiple equilibria. With multiple equilibria the restriction $P = \Psi(\theta, P)$ does not define a unique vector $P$ but a set of vectors. In this case, the MLE can be defined as:
>
> $$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \left\{ \sup_{P \in (0,1)^{N|X|}} Q_M(\theta, P) \text{ subject to: } P = \Psi(\theta, P) \right\} \quad (26)$$
>
> 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 $\theta$ we have to compute all the vectors $P$ which are an equilibrium associated with $\theta$ and then select the one with 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 $\Psi$ for different values of $P$ increases very importantly. These problems motivate the pseudo likelihood estimators we develop in the following subsections.

While the mathematical literature supports Aguirregabiria and Mira (Econometrica, 2007, pp. 1-53) in their assessment of the tractability of computing all equilibria of a game, it does not support their other claims, nor do they cite anything in the numerical mathematics literature consistent with their dismissal of mathematical programming methods. In particular, their key assertion that the "we have to compute all the vectors $P$ which are an equilibrium associated with $\theta$" is contradicted by the constrained optimization literature going back to the 1960's. In fact, it was not even true for the SUMT method.

The developments in nonlinear programming over the past 40 years shows that the description of the nonlinear programming literature explicitly contained in Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) and implicitly made elsewhere is false. Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) used these false assertions as the motivation for developing their pseudo maximum likelihood methods. Perhaps their techniques have some value, but their criticisms of the nonlinear programming approach to estimation, the approach we take in this paper, rest on false portrayals of the numerical analysis literature.

While the negative portrayals of numerical optimization commonly seen in economics are false, there is always the possibility that economics problems have unique features that

make possible ad hoc methods superior to standard numerical optimization methods. It is impossible to argue that standard numerical optimization methods will always do better than some ad hoc method which takes advantage of some special structure. However, we show below that standard numerical optimization methods, without any tweaking, can solve problems similar to those that have appeared in the economics literature and do so faster than methods that incorporate the special features of the economics model. These examples will serve two purposes: first, to show that standard numerical optimization methods are practical and efficient, and second, describe the critical features of economic models that will make these methods applicable.

The basic idea in this work is that the best approach to "Reducing the Computational Burden of Structural Estimation" is to consider methods developed by computational scientists and mathematicians and used by scientists and engineers to solve the key numerical tasks. The mathematical theory behind the methods we use clearly shows that these methods are asymptotically superior in terms of the rates of convergence. Furthermore, our use of standard software and hardware available to anyone for free to compute efficient estimates show that these methods will often be more effective than two-stage techniques in making "structural models accessible to a larger set of researchers".

## 3. MPEC Approach

We first formulate the general MPEC method. Suppose that an economic model is described by parameters $\theta$, and that in equilibrium the observable economic variables, $X$, follow a stochastic process parameterized by a finite vector $\sigma$ together with $\theta$. One example is dynamic programming problems where $\theta$ are the parameters for costs, benefits, laws of motion, and stochastic shocks, and where $\sigma$ is the policy function of a decisionmaker in an environment describe by the structural parameters $\theta$. In general, $\sigma$ will depend on $\theta$ through a set of equilibrium conditions such as first-order conditions, market balance conditions, etc., which are expressed in the system of equations[4] denoted by

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

Econometrics denotes the likelihood of a data set, $X$, conditional on parameter $\theta$ by $L(\theta, X)$, and maximum likelihood estimation solves

$$\max_{\theta} L(\theta, X).$$

This approach disguises many critical details by not explicitly expressing the dependence of $L$ on $\sigma$. In many ways, $\sigma$ is more directly related to the likelihood than some components

[4]More generally, equilibrium will be a system of complementarity conditions. In this paper, we stay with the simpler case where equilibrium is defined by a set of equations.

of $\theta$. For example, in a life-cycle problem, the likelihood of an observation depends on the consumption and labor decisions, not on, for example, the elasticity of labor supply. Elasticities affect likelihood only indirectly through their impact on decision rules.

To capture these details, we construct an *augmented likelihood function*, $\mathcal{L}(\theta, \sigma, X)$, which explicitly expresses the dependence of the likelihood on $\sigma$. This function makes no requirement that $\theta$ and $\sigma$ are consistent with the equilibrium conditions of the economic model. For example, in a life-cycle model, the policy function $\sigma$ together with the components of $\theta$ related to exogenous shock processes defines a stochastic process for the observables. A stochastic process is well-defined even if $\sigma$ and $\theta$ are not consistent with equilibrium. An augmented likelihood function allows us to treat $\sigma$ and $\theta$ independently when we compute the likelihood.

When we compute the maximum likelihood estimate for $\theta$ we want the $\sigma$ we compute to be consistent with $\theta$. Therefore, maximum likelihood estimation is the constrained optimization problem

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

The two formulations are mathematically equivalent. Let $\Sigma(\theta)$ be the set of all $\sigma$ such that $0 = G(\theta, \sigma)$; $\Sigma$ is implicitly defined by

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

Then the likelihood and augmented likelihood functions satisfy

$$L(\theta, X) = \mathcal{L}(\theta, \Sigma(\theta), X).$$

This equivalence also shows that the augmented likelihood function is a more fundamental expression of the problem since one can define $L$ in terms of $\mathcal{L}$ and $\Sigma$, but not vice versa.

This is a very general formulation. The MPEC approach does not require that equilibrium be defined as a solution to a fixed-point equation. We do not need to specify an algorithm for computing $\sigma = \Sigma(\theta)$; it is doubtful that we could do better than a good solver would do. 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 convergence properties than GJ and GS typically do. Using a direct optimization approach allows one to take advantage of the best available methods from the numerical analysis literature.

One advantage of the MPEC approach is obvious: our augmented likelihood function, $\mathcal{L}(\theta, \sigma, X)$, uses *only single-valued functions*! When $\Sigma(\theta)$ is multivalued, $L(X, \theta)$ is mul-

tivalued and the maximum likelihood optimization problem is difficult to solve if not intractable.

The other obvious point is that there is nothing unique about the likelihood function as an objective. The MPEC approach could be applied using any loss (or gain) function, such as least squares, weighted GMM, simulated maximum likelihood, etc., as the objective. The sections below will present a variety of applications of the MPEC approach.

**3.1. The Geometry of Constrained Estimation.** We next illustrate the basic ideas of NFXP and MPEC estimation in a graphical form that will help make clear various distinctions. Figure 1 shows what the likelihood function may appear to the NFXP solver. The likelihood function could be discontinuous due to the presence of multiple solutions at some values of $\theta$, the structural parameters. This will create difficulties for NFXP.

There are several ways to solve the maximization problem and avoid the difficulties associated with NFXP. The main problem with NFXP for estimating games and other problems where multiplicity of equilibria is a possibility is that it requires that each iterate be a $(\theta, \sigma)$ pair that satisfies all equilibrium conditions. Figure 2 displays a homotopy approach to the problem. If one began with a feasible point and used differential equation methods (similar to homotopy in style) that stayed confined to the feasible set but always moved to higher likelihood values then the path would rise to a local maximum. Therefore, a local maximum (which is all NFXP can guarantee) could be attained without computing all the structural models associated with a particular $\theta$.

Figure 3 illustrates the MPEC approach. We create an objective function that depends explicitly on both $\theta$ and $\sigma$. The constraints that equilibrium conditions impose on the $(\theta, \sigma)$ vector are modeled as manifolds in $\theta - \sigma$ space. The maximum likelihood problem then becomes one of maximizing the augmented likelihood function subject to these constraints. Nothing is discontinuous unless the constraints or likelihood function are. Therefore, the MPEC approach does not introduce discontinuities and nondifferentiabilities that are not part of the structure of the problem. Both NFXP and the homotopy-style method enforce equilibrium at each $(\theta, \sigma)$ examined, but standard solvers do not enforce feasibility at each iterate. This is one reason why they are usually much faster than methods that do enforce feasibility. This extra degree of freedom is made clear in Figure 3 where the feasible points are distinct from infeasible points.

**3.2. Inference Computations.** While the MPEC approach is equivalent to alternative approaches, implementing aymptotic inference methods is more complex with the MPEC approach. Computing standard errors for maximum likelihood estimates require the computation of the likelihood function, $L_{\theta\theta}$. This is a direct computation if one has a closed-form

# Figure 1

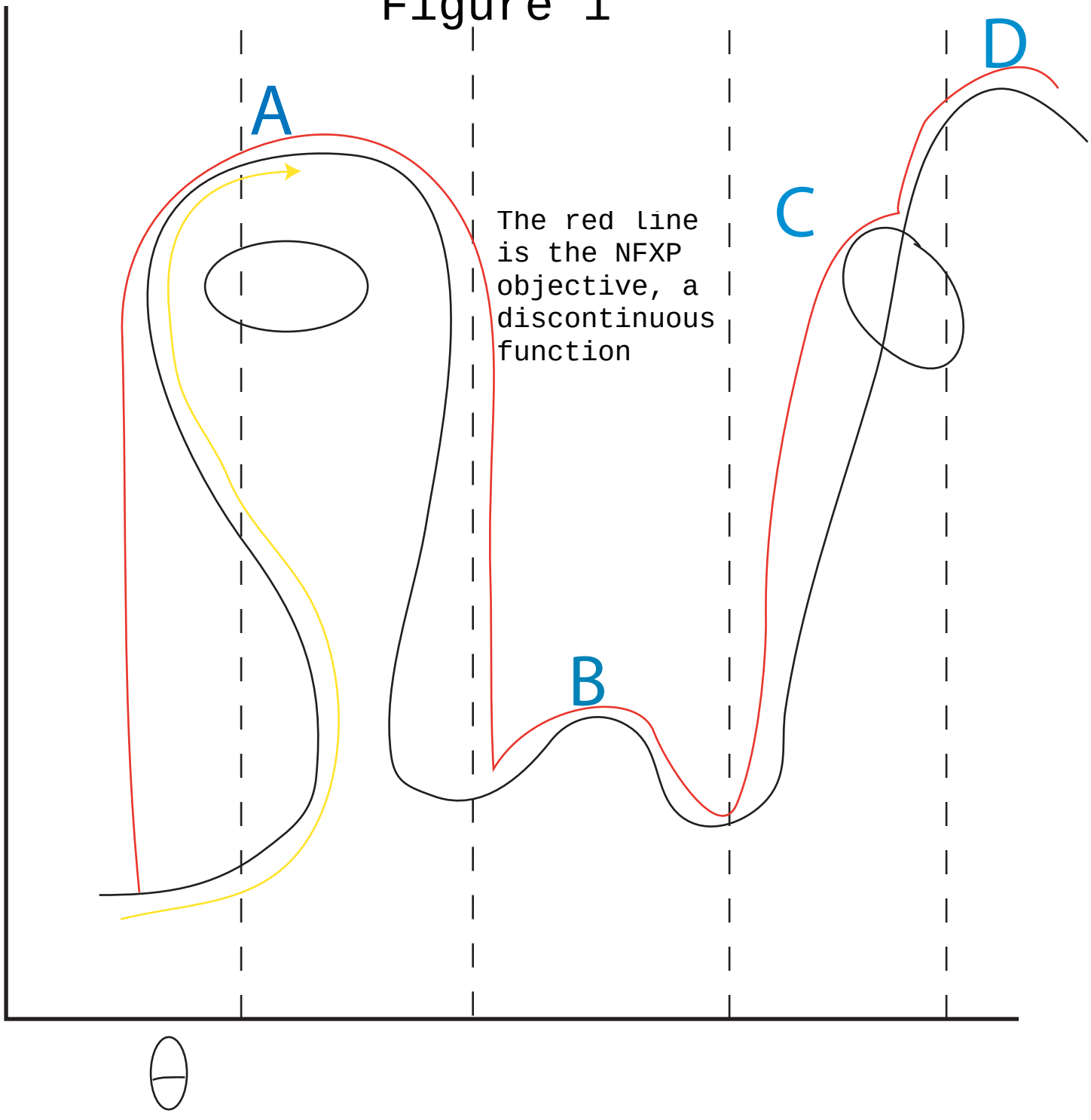
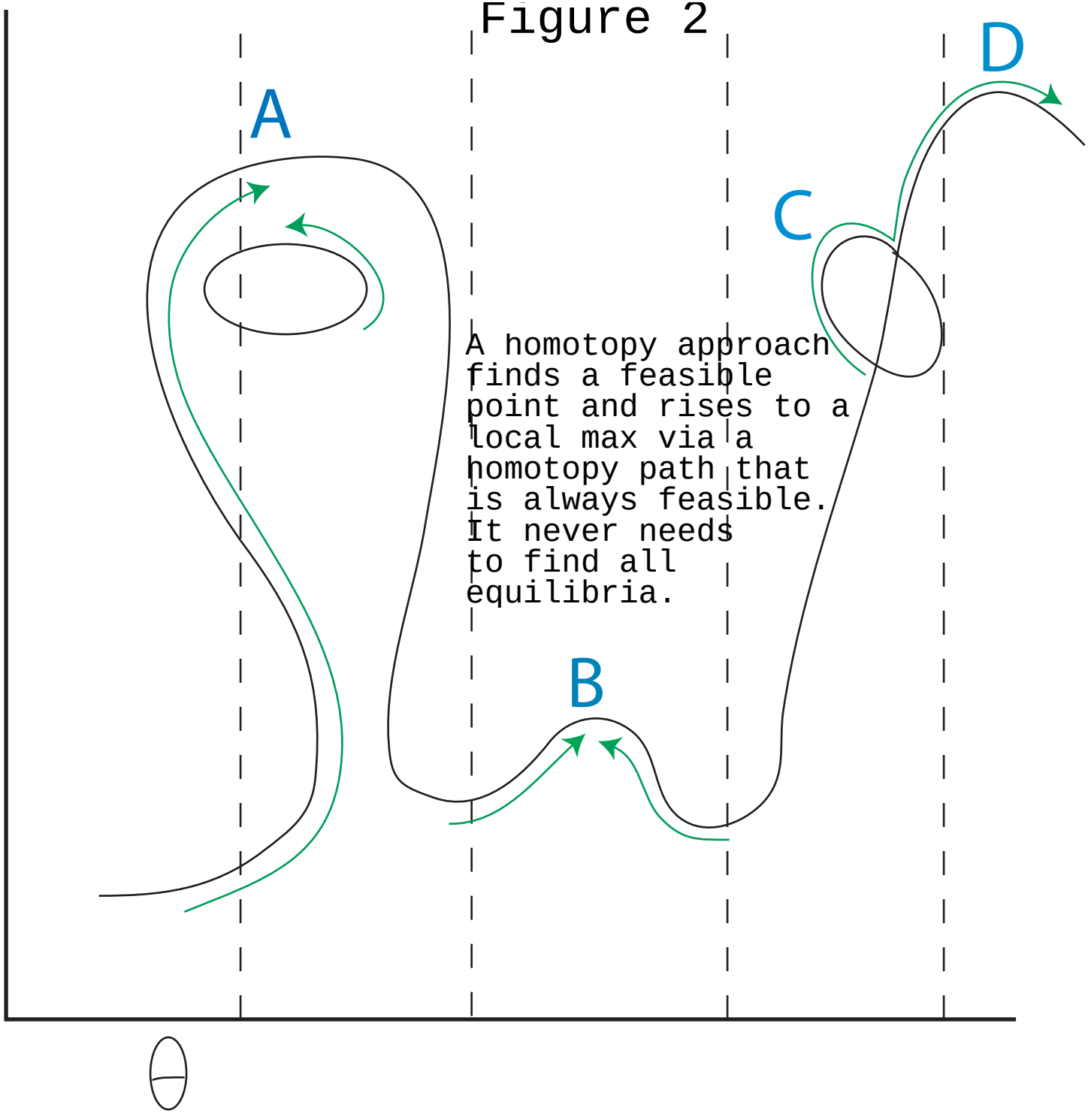
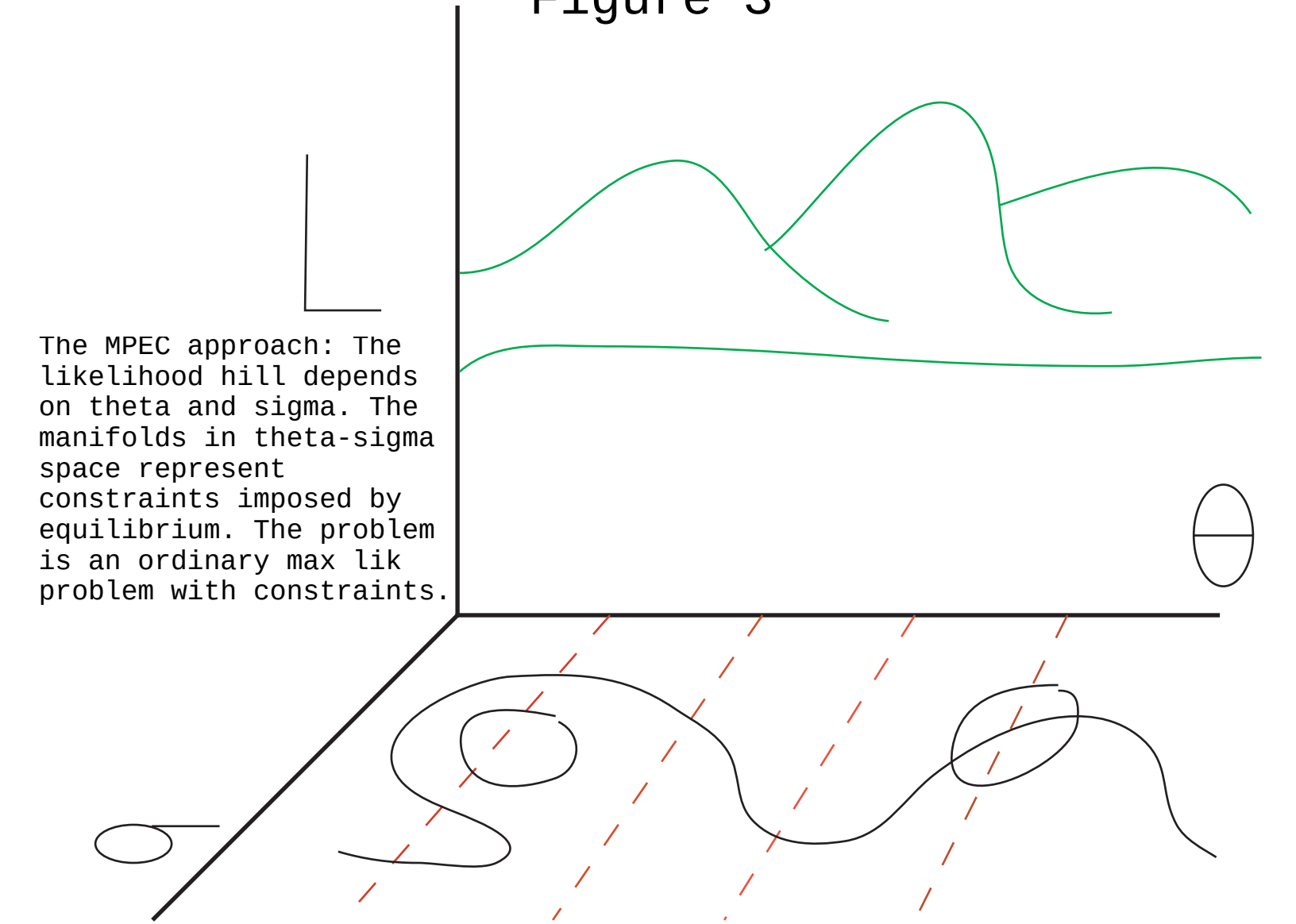

A
B
C
D
The red line is the NFXP objective, a discontinuous function

Figure 2


A
B
C
D
A homotopy approach finds a feasible point and rises to a local max via a homotopy path that is always feasible. It never needs to find all equilibria.

# Figure 3



The MPEC approach: The likelihood hill depends on theta and sigma. The manifolds in theta-sigma space represent constraints imposed by equilibrium. The problem is an ordinary max lik problem with constraints.

solution for the likelihood, or a numerical procedure as in the NFXP method. To use MPEC results to compute standard errors, we need to work through the implicit construction of the likelihood function. Recall that the likelihood and augmented likelihood functions are related by $L(\theta) = \mathcal{L}(\theta, \Sigma(\theta))$. To compute $L_{\theta\theta}$, we need to compute the Jacobian of $\Sigma$ with respect to $\theta$. The details of this computation and proofs concerning its validity are contained in Silvey (1958) where general constrained estimation problems are analyzed. Since this step in the MPEC approach is standard statistics, we need not spend time discussing it.

**3.3. Nonlinear Optimization Analogues.** The current estimation methods, NFXP and PML, correspond to alternative methods for solving nonlinear optimization problems. To see this clearly, consider problem

$$\begin{array}{ll} \max\limits_{\theta,y} & f(\theta, y) \\ \text{s.t.} & y = g(y, \theta) \end{array}$$

The NFXP is essentially a nonlinear substitution of variables method. First, define $Y(\theta)$ by $Y(\theta) = g(Y(\theta), \theta)$. Then substitute out the $y$ variables in the objective to get

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

In NFXP, $Y(\theta)$ is computed numerically for each value of $\theta$ if $Y(\theta)$ cannot be computed in closed form.

PML is essentially a Gauss-Seidel method. Suppose that you have a guess for $y$, say $y^i$. Then the next guesses for $y$ and $\theta$ are given by

$$\begin{aligned} \theta^{i+1} &= \arg\max_{\theta} f\left(\theta, y^i\right) \\ y^{i+1} &= g\left(y^i, \theta^{i+1}\right) \end{aligned}$$

The convergence of any Gauss-Seidel method is at best linear. More problematic is the fact that it is difficult to prove convergence, even local, of a Gauss-Seidel method.

In contrast, the methods we will use below are quadratically convergent in theory since they are based on Newton's method. Newton-style methods are superior since they use derivative information that is not used in Gauss-Seidel methods. Implementations of Newton-style methods may fall short of this ideal, but convergence is still far better than superlinear. Therefore, nonlinear substitution and nonlinear Gauss-Seidel methods are not considered viable alternatives in the nonlinear programming literature. In very large problems, various iterative methods are invoked to solve the linear equations that define a

Newton step, but these iterative methods for linear problems are better behaved globally than nonlinear Gauss-Seidel methods, and the Newton character still implies superlinear convergence for the overall algorithm.

## 4. MPEC Applied to a Simple Demand Example

We first apply the basic idea to a familiar example. Suppose that you have data on demand, $q$, and price $p$, but that demand is observed with error $\varepsilon$. Hence, true demand is $q - \varepsilon$. Suppose that you assume a parametric form for the utility function $u(c;\beta)$ where $\beta$ is a vector of parameters. Economic theory says that $u_c(c;\beta) = u_c(q-\varepsilon;\beta) = p$, providing a structural relation that can be used to estimate $\beta$.

Consider first the way you were taught how to do this. For example, you might assume a functional form for demand, such as

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

In this case, you were told to compute the demand function

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

which in turn implies that the $i$'th data point satisfies

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

for some $\varepsilon_i$. To estimate $\beta$, you were told to minimize the sum of squared errors, as in

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

The closed-form demand function approach has severe limitations. For example, suppose you wanted to examine the utility function

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

which is as reasonable a utility function as the quadratic utility function. The problem is that the first-order condition implies

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

an equation which has no closed-form solution expressing demand in terms of price. The closed-form approach immediately breaks down and it is impossible to continue!

If a closed-form solution does not exist, one can still proceed by dealing with the first-order condition directly. The first-order condition contains all the information you can have.

The demand function is just another way of representing the same information, but is often difficult, if not impossible, to express in a compact and efficient manner. It is unnecessary as well as absurd to limit yourself to the few special cases for which there are closed-form solutions to express the demand function, or any other function that arises.

The proper way to proceed is to realize that all you want to do in least squares estimation is to find the errors that are smallest in terms of their sum of squares but can be consistent with the critical structural equation. For our consumption demand model, this is the problem

$$\begin{array}{ll} \min\limits_{\varepsilon_i,\beta} & \sum\limits_{i=1}\varepsilon_i^2 \\ \text{s.t.} & u_c\left(q_i-\varepsilon_i;\beta\right)=p_i \end{array}$$

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

$$\begin{array}{ll} \min\limits_{c_i,\varepsilon_i,\beta} & \sum\limits_{i=1}\varepsilon_i^2 \\ \text{s.t.} & 1-2\beta c_i=p_i \\ & q_i=c_i+\varepsilon_i \end{array}$$

and in the case of the degree-six utility function, the problem becomes

$$\begin{array}{ll} \min\limits_{c_i,\varepsilon_i,\beta} & \sum\limits_{i=1}\varepsilon_i^2 \\ \text{s.t.} & 1-\beta\left(2c_i+4c_i^3+6c_i^5\right)=p_i \\ & q_i=c_i+\varepsilon_i \end{array}$$

Even when you can solve for the demand function, it is not clear you want to. Consider the case

$$\begin{aligned} u\left(q\right) &= q-\beta_1 q^2-\beta_2 q^3-\beta_3 q^4-\beta_4 q^5 \\ u'\left(q\right) &= 1-2\beta_1 q-3\beta_2 q^2-4\beta_3 q^3-5\beta_4 q^4 \end{aligned}$$

The demand function is one of the four solutions to the quartic polynomial in the first-order condition, where the four solutions are

$$\begin{aligned} q &= -\frac{c}{4d}\pm\frac{\sqrt{D+F/2}}{2}\pm\frac{\sqrt{-D+F-G}}{2} \\ G &= \frac{c^3-4bcd+8ad^2}{\sqrt{8}d^3\sqrt{2D+F}},\ F=\frac{c^2}{2d^2}-\frac{4b}{3d} \\ D &= \frac{2^{1/3}B}{3C^{1/3}d}-\frac{C^{1/3}}{3\ 2^{1/3}d},\ C=A\sqrt{A^2-4B^3} \\ B &= b^2-3ac+12d \\ A &= 2b^3-9abc+27c^2+27a^2d-72bd \end{aligned}$$

where

$$a = 2\beta_1,\ b = 3\beta_2,\ c = 4\beta_3,\ d = 5\beta_4$$

To compute demand, one needs to compute all four solutions and then pick the one that makes economic sense, such as being a positive real number, satisfying the second-order optimality condition, and being globally optimal.

Since the demand function involves many floating point operations, some of which, such as roots, are expensive to compute, it is much slower to compute than the first-order conditions. Also, the least-squares problem is much more nonlinear when we use the explicit demand function, involving cube and square roots, than the simper polynomial terms that arise in the constrained optimization approach. Mathematically, the MPEC approach includes the standard approach. Suppose there is a closed form solution for demand, $D(p, \beta)$, with parameters $\beta$. In this case, one way to implement the MPEC method is to solve

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

where the constraints $c_i = D(p_i, \beta)$ are just algebraically altered versions of $u_c(c_i; \beta) = p_i$. Which version is better depends on practical issues such as cpu time and the amount of nonlinearity. Therefore, the real task is writing the constrained optimization problem in a manner that minimizes the total computational burden given the finite speed and precision of computers.

These simple examples show that the habit of restricting models to cases with closed-form solutions is unnecessary. There is no reason for economists to impose this burden on themselves instead of writing down the models they really want to analyze and solving them.

## 5. MPEC applied to Zurcher

We apply the MPEC method to the bus repair model analyzed in Rust (1994). We use the Rust model since it is often used to exposit the ideas of maximum likelihood estimation of dynamic models and to evaluate alternative methods.

The bus repair problem considers the decisions faced by a repairman who must decide whether to perform extensive repairs on a bus when it comes into the bus barn and return it to excellent shape, or to implement less costly activities. The state variable is, $x_t$, the accumulated mileage at time $t$ since the last engine replacement. Let $c(x, \theta^c)$ be the expected

per period operating costs, where $\theta^c$ is a vector of parameters of the cost function. Some of these costs are not observed by the econometrician; hence, $\theta^c$ is to be estimated from the observations of when the bus is repaired. Rust assumes that the mileage travelled by a bus during one month is exponentially distributed with parameters $\theta^p$. In each period, a bus arrives in state $x$ and Zurcher decides between (i) "normal maintenance" incurring costs $c(x, \theta^c)$, and (ii) replace the bus engine, with a cost equal to $RC$ net of any scrap value for the old engine, and spend $c(0, \theta^c)$ on maintenance costs for a bus with a new engine. The data is the time series $(x_t^i, d_t^i)_{t=1}^T$, where $x_t^i$ is state of bus $i$ examined in period $t$ and $d_t^i$ is the choice made for that bus. The data set may also merge data from different busses. The objective is to find the parameter values that maximize the likelihood of that data.

Zurcher chooses a replacement policy to minimize the expected discounted maintenance costs. The replacement decision is denoted by $d_t^i$ where $d_t^i = 1$ if the engine is replaced and $d_t^i = 0$ otherwise. Given the data $(x_t^i, d_t^i)_{t=1}^T$, where $x_t^i$ is state and $d_t^i$ is the decision (or choice) associated with $x_t^i$, we want to infer the unknown parameter vector $\theta$ by maximizing the *likelihood function* $L(\theta)$

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

where the *conditional choice probability*, $P(d|x.\theta)$ is given by the *multinomial formula*

$$P(d|x, \theta) = \frac{exp\{u(x, d, \theta) + \beta V_\theta(x, d)\}}{\sum_{d' \in D(x)} exp\{u(x, d', \theta) + \beta V_\theta(x', d)\}}, \tag{2}$$

and the expected value function $V_\theta$ is the fixed point of the contraction mapping $T_\theta(V)$ and solves the Bellman equation

$$V(x, d) = \int_{x'=0}^{\infty} \log \left[ \sum_{d' \in D(x')} exp\{u(x', d', \theta) + \beta V(x', d')\} \right] p(dx'|x, d, \theta). \tag{3}$$

We want estimate the unknown parameter vector $\theta = (\theta^c, RC, \theta^p)$ by solving the constrained optimization problem

$$\begin{aligned} &\max_{\theta, V} \quad \prod_{i=1}^{M} \prod_{t=2}^{T} P(d_t^i|x_t^i, \theta) p(x_t^i|x_{t-1}^i, d_{t-1}^i, \theta) \\ &\text{s. t.} \qquad V = T_\theta(V) \end{aligned} \tag{4}$$

This is not possible since $V$ is a function on $R$, implying that the constraint is a functional equation. Therefore, we need to approximate the value function in a finite-dimensional manner. We take the same approach as in Rust and discretize the state space. Let $N$ be the number of states for $x$, and $Z$ the set of states. This approach requires that the data

also live on the same finite grid. Therefore, the data is transformed, replacing each $x_t^i$ with the nearest value in $Z$, which we denote $\hat{x}_t^i$. Hence, the computational approximation solves

$$\begin{aligned} \max_{\theta,V} \quad & \prod_{i=1}^{M} \prod_{t=2}^{T} P(d_t^i|\hat{x}_t^i,\theta) p(\hat{x}_t^i|\hat{x}_{t-1}^i, d_{t-1}^i, \theta) \\ \text{s. t.} \quad & V(z) = T_\theta(V)(z), \; z \in Z \end{aligned} \tag{5}$$

**5.1. A Numerical Example.** We consider a specific example of Rust's bus engine replacement model. We choose the state space $Z = \{1, \ldots, N\}$ and assume the cost function $c(x, \theta)$ is quadratic, i.e., $c(x, \theta^c) = \theta_1^c x + \theta_2^c x^2$. We do not estimate the discount factor $\beta$ and let $\beta = 0.95$. The unknown parameters to be estimated are $\theta^c$, $RC$ (scrap value) and $\theta^p$, parameters defining the Markov transition probabilities. We simulate a time series data for $T$ time periods ($T = 10^3, 10^4$) by using the following parameter values: $\beta = 0.95$, $RC = 1.217$, $\theta^c = (0.06, 0.0005)$, and $\theta^p = (0.35, 0.60, 0.05)$. We submitted the problem, coded in AMPL, to the SNOPT solver via NEOS. The computer used was an ordinary desktop computer. The outputs are given below.

We first estimate three parameters as in the Rust example. Rust estimated the Markov transition parameters in a first stage and then used those probabilities in his NFXP estimation of the remaining parameters. Table 1 displays the results.

Table 1: Three-Parameter Estimates

| | | Estimates | | | CPU | Major | Evals* | Bellman |
|---|---|---|---|---|---|---|---|---|
| $T$ | $N$ | $RC$ | $\theta_1^c$ | $\theta_2^c$ | (sec) | Iterations | | 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 | | | | | | | | |

A statistically more efficient approach is to estimate all parameters, including the Markov transition parameters, in one maximum likelihood estimation procedure since deci-

sions help reveal what Zurcher knows about the probabilities. Table 2 reports those results.

Table 2: Five-Parameter Estimates

| | | Estimates | | | | | CPU | Major | Evals | Bellman |
|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $N$ | $RC$ | $\theta_1^c$ | $\theta_2^c$ | $\theta_1^p$ | $\theta_2^p$ | (sec) | Iterations | | error |
| $10^3$ | 101 | 1.107 | 0.039 | 0.0030 | 0.723 | 0.262 | 0.50 | 111 | 137 | 6.1E−12 |
| $10^3$ | 201 | 1.140 | 0.055 | 0.0015 | 0.364 | 0.600 | 1.14 | 109 | 120 | 1.3E−09 |
| $10^3$ | 501 | 1.129 | 0.050 | 0.0019 | 0.339 | 0.612 | 3.39 | 115 | 127 | 2.5E−11 |
| $10^3$ | 1001 | 1.144 | 0.056 | 0.0014 | 0.360 | 0.608 | 7.56 | 84 | 116 | 5.1E−12 |
| $10^4$ | 101 | 1.236 | 0.052 | 0.0016 | 0.694 | 0.284 | 0.50 | 76 | 91 | 5.3E−11 |
| $10^4$ | 201 | 1.257 | 0.060 | 0.0010 | 0.367 | 0.053 | 0.86 | 85 | 97 | 3.6E−11 |
| $10^4$ | 501 | 1.252 | 0.058 | 0.0012 | 0.349 | 0.596 | 2.73 | 83 | 98 | 2.9E−10 |
| $10^4$ | 1001 | 1.256 | 0.060 | 0.0010 | 0.370 | 0.586 | 19.12 | 166 | 182 | 3.2E−10 |

A variety of points are clear. First, the problems are solved quickly, usually around a second and requiring significantly more time only with the larger data sets and the finer grid in the dynamic programming approximation. We also want to compare the speeds with those reported in Rust. This comparison of speed cannot rely on clock time since different computers have different capabilities, particularly when they are of such different vintages. The key measure of time is the number of times the objective functions and constraints need to be computed. We see that very few evaluations are necessary, even when estimating five parameters. In NFXP, solving one dynamic programming problem would require dozens of evaluations of the Bellman equations (our constraints). Due to the use of finite differences for derivatives, each outer iteration requires solving four dynamic programming problems and evaluating the likelihood four times when estimating three parameters. These details all depend on the starting guess. We initially assumed that the value function is zero. If NFXP used that starting guess, our MPEC method finishes the whole problem before NFXP has solved even one dynamic programming problem! All methods benefit from good initial guesses. The key fact is that NFXP solves many dynamic programming problems, whereas the MPEC method uses far fewer evaluations of the likelihood and the Bellman equations since it solves only one dynamic programming problem.

Second, notice how the estimates differ as we refine the approximation of the dynamic programming problem. The true state space is continuous. We discretize the state space as did Rust, but we examine a variety of discretizations. The cases where we used 1000 states are ten times as fine in their discretizations as in Rust. It appears that coarser discretizations produce nontrivial errors in the estimated parameters.

**5.2. Parametric Bootstrap.** Statistical inference relies on the computation of a standard error. Above we described how one could compute the Hessian of the likelihood function at the point estimate. Some prefer bootstrap methods for computing standard errors. Boostrapping may seem difficult for the MPEC method since we would need to resolve the estimation problem many times. We show that this is not a problem.

Since the data has a time series character, we do not use a resampling approach to bootstrapping. Instead, we used a parametric bootstrap method; that is, we took the solution to the policy function implied by our point estimate and used it to generate several synthetic data samples of the same size as our original problem. This gives us some idea about precision of these estimates.

We generated 20 data sets of 1000 points. Each data set takes the point estimates for the parameters, computes the implied decision rule, and simulates the data process. We then estimated the five parameters using 1001 grid points in the dynamic programming approximation. Note that the running times were all similar for the various data sets. Also, many data sets implied corner solutions to the cost function, where we imposed monotonicity and convexity conditions.

Table 3: Maximum Likelihood Parametric Bootstrap Results

| | Estimates | | | | | | CPU | Major | Evals | Bell. EQ |
|---|---|---|---|---|---|---|---|---|---|---|
| | $RC$ | $\theta_1^c$ | $\theta_2^c$ | $\theta_1^p$ | $\theta_2^p$ | $\theta_3^p$ | (sec) | Iter | | Error |
| **Mean** | 1.14 | 0.037 | 0.004 | 0.384 | 0.587 | 0.029 | 0.54 | 90 | 109 | 8.2E−09 |
| **S.E.** | 0.15 | 0.035 | 0.004 | 0.013 | 0.012 | 0.005 | 0.16 | 24 | 37 | 1.7E−08 |
| **Min** | 0.95 | 0.000 | 0.000 | 0.355 | 0.571 | 0.021 | 0.24 | 45 | 59 | 1.2E−13 |
| **Max** | 1.46 | 0.108 | 0.012 | 0.403 | 0.606 | 0.039 | 0.88 | 152 | 230 | 5.7E−08 |

**5.3. Comparisons with NFXP.** We next compare NFXP with the MPEC approach in several dimensions. We first consider the general differences, and then discuss the differences between how Rust implemented NFXP and how we implemented the MPEC approach.

First, if solution time is large relative to likelihood evaluation time, the MPEC approach results in a far faster algorithm. We do not solve the dynamic programming problem for each $\theta$ guess. For each $\theta$ that is considered in NFXP, there are two steps: solving the dynamic programming problem with high accuracy and evaluating the likelihood function. Furthermore, this is repeated for nearby values of $\theta$ if finite difference methods are used to compute derivatives. In contrast, for each $\theta$ considered in MPEC, the Bellman equation is *evaluated* once, not *solved*. Value function iteration will evaluate the Bellman equation

many times in NFXP each time a dynamic program is solved. Essentially, MPEC nearly eliminates the time spent on solving the dynamic programming problem.

Even if one had a closed-form solution for the dynamic programming problem, it is not clear that it is a good idea to use it. Nonlinear elimination of variables reduces the number of unknowns but possibly at the cost of increasing the nonlinearity of the objective function. If the equilibrium equations are sparse, as is true with the Zurcher dynamic programming problem and many other problems, then little time is saving from substituting out variables. Actually, it is often easier to solve large optimization problems! Substituting out $m$ variables from $n$ squeezes all the nonlinearities of those $m$ dimensions into the remaining $n - m$ dimensions and can easily create a highly nonlinear objective. In optimization, it is nonlinearity, not dimensionality, that makes a problem difficult.

The second kind of advantage of MPEC is the far greater flexibility and ease in implementation. Software implementations are much easier since it makes derivatives much easier to compute, and allows for the direct application of methods that produce efficient analytic derivatives. This would be very difficult to do in NFXP, leaving NFXP to rely on the inferior and inefficient finite difference derivatives. Also, this approach encourages one to experiment with many solvers to find the best one.

**5.4. Comparison with Rust's Implementation of NFXP.** Some of the reason for our superior results is that we use different software than Rust does. Also, the running times he reports cannot be compared to ours since computer technology has significantly improved. However, some comparisons can be made that are independent of technology. We next run through the points made in Rust (1987) about the advantages of Gauss, and compare his Gauss implementation of NFXP with our AMPL implementation of MPEC.

**Ease of use.** Rust used Gauss "because: 1. the GAUSS language 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 just as easy to use. All solvers include linear algebra routines in Lapack. AMPL does not have matrix notation, but its summation notation for matrices and tensors is more flexible than a matrix language.

**Vectorization.** Rust: "Efficient use of GAUSS requires the user to "vectorize" a program in a manner very similar to efficient vectorization on large-scale vector supercomputers."

MPEC: All vectorization is done automatically in AMPL. The user just expresses the problem using a modeling language with a user-friendly syntax. No need to think about

vectorization.

**Optimization Method.** Rust: Outer iteration uses BHHH for a while then switches to BFGS, where the user chooses the switch point.

MPEC: The algorithms contained in the solvers of the past 30 years are far superior to these methods. They are faster, and do not require user intervention regarding any switching points.

**Derivatives.** Rust: "The NFXP software computes the likelihood and its derivatives numerically in a subroutine. This implies that we can numerically compute the likelihood 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: We 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.

MPEC: We use only Newton-style methods. They are faster than contraction mapping methods. This is particularly important if $\beta$ is close to 1, representing short, but realistic, time periods. Rust starts with the contraction iterations because of instabilities using Newton's method, but this problem arises only with the pure Newton method. However, Newton's method works far better if combined with either a trust region or linesearch method, as is the practice now in all good optimization software. Moreover, Newton's method combined with one of these strategies will always converge to the true solution of the Bellman equations.

The MPEC approach could be implemented in Gauss using the CML module. However, the CML module does numerical derivatives (unless the user provides derivatives) and uses only one algorithm. The big advantages of AMPL are its implementation of automatic differentiation and it giving us access almost all of the state-of-the art solvers.

Some of the features used in our implementation of the MPEC approach could be implemented in NFXP. For example, one could use the version of Newton's method used in our MPEC implementation to reliably and rapidly solve the dynamic programming problem solved at each stage in NFXP. However, no implementation of NFXP can avoid the problem of resolving the dynamic programming problem at each guess of $\theta$. This feature of NFXP

makes it particularly inappropriate for application to games, a topic to which we next turn.

## 6. The MPEC Approach to games

NFXP cannot be used to estimate data from games except for very special cases. Suppose that the game has parameters $\theta$ representing payoffs, probabilities, and whatever else is not observed directly by the econometrician. Let $\sigma$ denote the equilibrium strategy given $\theta$. Suppose that $\sigma$ is an equilibrium if and only if

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

for some function $G$.[5]. Suppose that the augmented likelihood function of a data set $X$ is $\mathcal{L}(\sigma, \theta, X)$. Therefore, the maximum likelihood estimator is the solution to

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

For each $\theta$, NFXP requires one to find all the $\sigma \in \Sigma(\theta)$ that solve $G(\sigma, \theta)$, compute the likelihood at each $\sigma \in \Sigma(\theta)$, and report the maximum. Finding all equilibria is an intractable problem unless one has special structure. Also, the resulting likelihood function will often be discontinuous if there are multiple equilibria, and possibly nondifferentiable even at points of continuity. Both of these problems will create difficulties for the outer loop in NFXP.

Aguirregabiria and Mira (*Econometrica*, 2007, pp. 1-53) propose an alternative approach to this problem using their pseudomaximum likelihood approach. However, the Gauss-Seidel nature of their iteration produces convergence problems. Therefore, their Monte Carlo examples only looked at equilibria that were stable under Best Reply (BR) iteration. Their use of a selection criterion limits the range of equilibria. In fact, there are trivial examples of games with unique equilibria but where BR iteration is unstable.

In contrast, MPEC just sends the problem to good optimization solvers. There is no need for assuming that the data comes from a particular kind of equilibrium. Multiplicity of equilibria will not create discontinuities or lack of differentiability in the MPEC approach. Multiple equilibria may produce multiple local solutions, but that is a standard problem in maximum likelihood estimation, and would also be a problem for the NFXP approach. The next figure illustrates the approach. The likelihood function is a smooth function of the parameters $\theta$ and the strategies $\sigma$, and the constraints define a set of points in $\theta - \sigma$ space. The result is a standard constrained optimization problem.

[5]In some games, equilibrium is defined by a set of complementarity conditions. We do not want to go into the details here, but that case is mathematically more challenging due to failures of conventional constraint qualifcations. Those difficulties can be addressed by methods from the MPCC literature.

We are not saying that solving this problem is easy. Of course, it will be more difficult and costly as the size and complexity of the game increases. The key advantage is that this approach is often just a conventional optimization problem with continuous objective and constraint functions. In contrast, the NFXP approach is impossible to implement for all but the simplest games.

**6.1. An Example of Games with Multiple Equilibria.** We illustrate these points with a simple example. Consider a Bertrand pricing game between two firms in several cities, where each city is one of four types. There are two products, $x$ and $y$, two firms with firm $x$ ($y$) producing good $x$ ($y$), and three types of customers. We assume that the equilibrium is the same in each city of the same type; let $p_{xi}$ ($p_{yi}$) be the price of good $x$ ($y$ ) in a type $i$ city. Let $Dx_1$, $Dx_2$, and $Dx_3$ be the demands for product $x$ by customer type 1, 2, and 3. Similarly for $Dy_1$, etc. Type 1 customers only want good $x$, and have a linear demand curve:

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

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 goods. Let $n_i$ be the number of type 2 customers in a type $i$ city. We assume that the two goods are imperfect substitutes for type 2 customers with a constant elasticity of substitution between the two goods and a constant elasticity of demand for a composite good. This implies the demand functions

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

where $\sigma$ is the elasticity of substitution between $x$ and $y$, and $\gamma$ is the elasticity of demand for a composite good. 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 $x$ in a type $i$ city, $R_x(p_{xi}, p_{yi})$, is $(p_{xi} - m)Dx(p_{xi}, p_{yi})$; $R_y$ is similarly defined. Let $MR_x$ be marginal profits

for good $x$; similarly for $MR_y$. The equations for $MR_x$ and $MR_y$ are

$$\begin{aligned} MR_x(p_{xi}, p_{yi}) &= A - p_{xi} + n_i \left( p_{xi}^{\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\gamma-\sigma}{\sigma-1}} \right)^{-1} \\ &\quad + (p_{xi} - m) \left( -1 + \frac{n_i(\sigma-\gamma)}{p_{xi}^{2\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{1+\frac{\sigma-\gamma}{\sigma-1}}} - \frac{n_i \sigma}{p_{xi}^{1+\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\sigma-\gamma}{\sigma-1}}} \right) \\ MR_y(p_{xi}, p_{yi}) &= A - p_{yi} + n_i \left( p_{yi}^{\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\gamma-\sigma}{\sigma-1}} \right) \\ &\quad + (p_{yi} - m) \left( -1 + \frac{n_i(\sigma-\gamma)}{p_{yi}^{2\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{1+\frac{\sigma-\gamma}{\sigma-1}}} - \frac{n_i \sigma}{p_{yi}^{1+\sigma} \left( p_{xi}^{1-\sigma} + p_{yi}^{1-\sigma} \right)^{\frac{\sigma-\gamma}{\sigma-1}}} \right) \end{aligned} \tag{6}$$

Equilibrium prices satisfy the system

$$\left. \begin{array}{l} MR_x(p_{xi}, p_{yi}) = 0 \\ MR_x(p_{xi}, p_{yi}) = 0 \end{array} \right\} \text{ for } i = 1, \ldots, 4. \tag{7}$$

Our example will assume that the four markets differ only in terms of population, $n_i$. The other parameters are common across markets; we assume they are

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

Given these parameter values, the marginal revenue functions are

$$\begin{aligned} MR_x(p_{xi}, p_{yi}) &= 50 - p_{xi} + (p_{xi} - 1) \left( -1 + n_i p_{xi}^{-6} P_i^{-3} - 3 n_i p_{xi}^{-3} P_i^{-1} \right) + n_i p_{xi}^{-3} P_i^{-1} \\ MR_y(p_{xi}, p_{yi}) &= 50 - p_{yi} + (p_{yi} - 1) \left( -1 + n_i p_{yi}^{-6} P_i^{-3} - 3 n_i p_{yi}^{-4} P_i^{-1} \right) + n_i p_{yi}^{-3} P_i^{-1} \\ P_i &= \sqrt{p_{xi}^{-2} + p_{yi}^{-2}} \end{aligned}$$

The key economic fact in our example is that each firm has two kinds of strategies it can follow. The *niche strategy* is to charge a high price and sell mainly to the relatively small number of customers that want only its good. The *mass market strategy* chooses a low price so that many of the type 2 customers will buy its product as well as the competing firm's product. The mass market customers will buy some of each, but the composition of their purchases is significantly price sensitive. Each firm has to choose between a high margin and low sales, or a low margin and high sales.

We choose the type 2 customer population in the four cities to be $(n_1, n_2, n_3, n_4) = (1500, 2500, 3000, 4000)$. For each city, we solve for the solutions to the above first-order conditions and check their second-order conditions. We also check global optimality for each firm in each potential equilibria. After performing all the necessary checks, we find

that the equilibrium is unique for type 1 and 4 cities:

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

The unique equilibrium in type 1 cities is for both firms to choose a niche strategy, whereas the unique equilibrium in type 4 cities is for both to pursue a mass market strategy. This is all very intuitive since the mass market is small in type 1 cities but large in type 4 cities.

The situation is more complex for type 2 and type 3 cities. The same procedure showed that there are two equilibria in type 2 cities and type 3 cities. In these equilibria, one firm chooses a niche strategy while the other chooses a mass market strategy. More precisely, the equilibria are:

$$\begin{aligned} \text{City 2:} \quad \left(p_{x2}^{I}, p_{y2}^{I}\right) &= (25.18,\ 2.19) \\ \left(p_{x2}^{II}, p_{y2}^{II}\right) &= (2.19,\ 25.18) \end{aligned}$$

$$\begin{aligned} \text{City 3:} \quad \left(p_{x3}^{I}, p_{y3}^{I}\right) &= (2.15,\ 25.12) \\ \left(p_{x3}^{II}, p_{y3}^{II}\right) &= (25.12,\ 2.15) \end{aligned}$$

Essentially, 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.

We assume the econometrician observes price data for $4K$ cities, with $K$ cities of each type; the data is denoted by $P = (p_{x,i}^k, p_{y,i}^k)_{i=1,2,3,4}^{k=1,..,K}$. Suppose that we want to estimate the unknown structural parameters $(\sigma, \gamma, A, m)$ as well as equilibrium prices implied by the data in all four cities. To make the problem nonsingular, we assume that each observed price, $p_{y,i}^k$, is contains measurement error; hence,

$$p_{z,i}^k = p_{z,i} + \varepsilon_i^k,\ z = x, y;\ i = 1, 2, 3, 4;\ k = 1, ..., K;\ \varepsilon \sim N(0, 50)$$

The augmented (log) likelihood function, $\mathcal{L}(p_x, p_y, \sigma, \gamma, A, m, P)$ is

$$\mathcal{L}(p_x, p_y, \sigma, \gamma, A, m, P) = -\sum_{k=1}^{K} \sum_{i=1}^{4} \left( (p_{xi}^k - p_{xi})^2 + (p_{yi}^k - p_{yi})^2 \right).$$

We want to maximize the likelihood function subject to the requirement that our estimates of the type-specific equilibrium prices, $(p_{xi}, p_{yi})_{i=1}^4$, and our estimates of the structural parameters, $(\sigma, \gamma, A, m)$, satisfy the first-order conditions (7). Because the firm's problems are not concave, we also need to worry about global optimality. The constrained optimization

formulation for this price game is

$$\begin{array}{ll}
\min_{(p_{xi},p_{yi},\sigma,\gamma,A,m)} & \sum_{t=1}^{K}\sum_{i=1}^{4}\left((p_{xi}^k - p_{xi})^2 + (p_{yi}^k - p_{yi})^2\right) \\
\text{subject to} & 0 = MR_y(p_{xi},p_{yi}) = MR_x(p_{xi},p_{yi}),\ i = 1,\ldots,4 \\
& p_{xi}, p_{yi} \geq 0,\ i = 1,\ldots,4 \\
& \forall p\left((p_{xi}-m)Dx(p_{xi},p_{yi}) \geq (p-m)Dx(p_j,p_{yi})\right), \quad i = 1,\ldots,4 \\
& \forall p\left((p_{yi}-m)Dy(p_{xi},p_{yi}) \geq (p-m)Dy(p_{xi},p_j)\right), \quad i = 1,\ldots,4
\end{array}$$

This is an example of a semi-infinite programming problem. We do not want to invoke the complexities of that approach here. Therefore, we will adopt the sampling approach to solving semi-infinite problems, and just require that the equilibrium prices $(p_{xi}, p_{yi})$ satisfy a finite number of global optimality conditions on a pre-specified grid of $J$ alternative prices, $\{p_1,\ldots,p_J\} = \{1, 2, ..., 30\}$:

$$\left.\begin{array}{rcl}
(p_{xi}-m)Dx(p_{xi},p_{yi}) & \geq & (p_j-m)Dx(p_j,p_{yi}) \\
(p_{yi}-m)Dy(p_{xi},p_{yi}) & \geq & (p_j-m)Dy(p_{xi},p_j)
\end{array}\right\} \text{ for } i = 1,\ldots,4,\ j = 1,\ldots,J.$$

The constrained optimization formulation for this price game now becomes

$$\begin{array}{ll}
\min_{(p_{xi},p_{yi},\sigma,\gamma,A,m)} & \sum_{t=1}^{K}\sum_{i=1}^{4}\left((p_{xi}^k - p_{xi})^2 + (p_{yi}^k - p_{yi})^2\right) \\
\text{subject to} & 0 = MR_y(p_{xi},p_{yi}) = MR_x(p_{xi},p_{yi}),\ i = 1,\ldots,4 \\
& p_{xi}, p_{yi} \geq 0,\ i = 1,\ldots,4 \\
& (p_{xi}-m)Dx(p_{xi},p_{yi}) \geq (p_j-m)Dx(p_j,p_{yi}),\ i = 1,\ldots,4,\ j = 1,\ldots,J: \\
& (p_{yi}-m)Dy(p_{xi},p_{yi}) \geq (p_j-m)Dy(p_{xi},p_j),\ i = 1,\ldots,4,\ j = 1,\ldots,J:
\end{array}$$

For illustration purpose, we assume that in this example the true parameter values are $(\sigma,\gamma,A,m) = (3, 2, 50, 1)$, and that the city types are $(n_1, n_2, n_3, n_4) = (1500, 2500, 3000, 4000)$. We assume that the equilibria in the four city types are

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

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 studied three different cases by varying the number of structural parameters to be estimated. We coded the constrained optimization problem in AMPL and submitted it to

the KNITRO (interior point option) solver via NEOS. We chose KNITRO to display that interior point methods could also be used for the MPEC approach to estimation.[6] In Case 1, we estimated only two parameters, $\sigma$ and $\gamma$ and fix $A_x = A_y = 50$ and $m_x = m_y = 1$ at the true parameter values. In Case 2, we estimated all six structural parameters but we also imposed the symmetry constraints on the two firms: $A_x = A_y$ and $m_x = m_y$. In Case 3, we estimated all six structural parameters without imposing the symmetry constraints. Since the estimation problem for this price game is not a convex program, we performed a multi-start procedure with 25 different starting points and chose the solutions with the lowest object value as our final solution.

The estimated structural parameters and the implied equilibrium prices are reported in Table 4. The results clearly show that in each of the three cases, the estimated parameters are very accurate compared to the true values. Although there are multiple equilibrium prices in this price game, the implied equilibrium prices from our solutions correctly identify the equilibrium strategy played by the two firms in the data generating process. The implied equilibrium prices are also very close to the true equilibrium prices. The computational burden was low; most problems with out of the 25 random restarts procedure are solved in seconds with an average time of around 3 seconds. Except for a few instances, most problems only require around 50 to 60 major iterations and on average, 85 function evaluations. For Case 1, 14 out of 25 different starting points converge to the global solution. For Case 2 (Case 3), 20 (21) out of 25 starting points converge to the global solution. The observation shows that solving a estimation problem with few structural parameters is not necessarily easier than that with more parameters.

[6]There are several more solvers that could be applied. In general, our impression is that SQP solvers (SNOPT, NPSOL, Filter, KNITROSQP) and interior point solvers (KNITROIP and IPOPT) will be successful, but that augmented Lagrangian solvers (MINOS and LANCELOT) will not be as reliable. The key problem is then possibility that a solver will evaluate the objective or a constraint at a point where the function is not defined, such as logarithm of a negative number. SQP and IP solvers generally impose simple bound constraints for all iterates whereas iterates of augmented Lagrangian solvers may not satisfy the constraints.

Table 4: Game Estimation Results

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

## 7. The MPEC approach and the Method of Moments

There are many estimation methods that employ iterative methods of essentially a Gauss-Jacobi and Gauss-Seidel fashion to compute estimates. In general, computational efficiency could be substantially improved by pursuing a constrained optimization approach utilizing far more sophisticated methods. Moment-based estimators are examples where this approach could be applied. We display the advantage of the MPEC approach for a method of moments estimator using the bus example above.

Suppose that $\theta$ are unknown structural parameters, $\sigma$ are decision rules and other endogenous variables, that $G(\sigma, \theta) = 0$ expresses equilibrium conditions, and that $M(\sigma, \theta)$ is a vector of moments implied by endogenous variables $\sigma$ and parameters $\theta$. As with the augmented likelihood function, we do not assume that $\sigma$ and $\theta$ together satisfy the equilibrium conditions when we write $M(\sigma, \theta)$. Let $M(X)$ be the corresponding data moments. Then the moment estimator can be found by solving

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

Moment estimators are useful even in contexts where we want to use maximum likelihood estimation. For example, the range of values for $\theta$ for which the log likelihood function is defined could be fairly narrow and difficult to find, making maximum likelihood estimation impractical. Moment-based estimation is more robust and not subject to this difficulty.

We apply the method of moments to the bus problem (5) in Section 5. For notational simplicity, we only consider the case with one bus. We first compute various moments of the data. Let $(x_t, d_t)_{t=1}^T$ be the vector of mileage and decision. In our example, we compute the data moment vector

$$M = (M_x, M_d, M_{xx}, M_{xd}, M_{dd}, M_{xxx}, M_{xxd}, M_{xdd}, M_{ddd})$$

where $M_x$ is the mean of $x$, $M_d$ is the mean of $d$, $M_{xx}$ is the variance of $x$, $M_{xd}$ is the covariance of $x$ and $d$, $M_{xxx}$ is the skewness of $x$, etc. Let

$$m = (m_x, m_d, m_{xx}, m_{xd}, m_{dd}, m_{xxx}, m_{xxd}, m_{xdd}, m_{ddd})$$

denote the vector of the corresponding moments implied by the model.

We next need to compute the moments predicted by the structural parameters $\theta$. Let $\Pi$ denote the transition matrix from state $(x, d)$ to $(x', d')$. $\Pi$ will depend on the dynamic programming decision rule $\sigma$ and structural parameters $\theta$. Let $\Pi = H(\sigma, \theta)$ denote that functional relation. This is an appropriate expression for the relation between $\Pi$ and $(\sigma, \theta)$ since $H$ is a direct computation once we fix $\sigma$ and $\theta$. Let $p = (p_{x,d})_{x \in Z, d \in \{0,1\}}$ denote the stationary distribution (which will be unique since all ergodic states communicate). Therefore, $p$ and $\Pi$ are the solutions to

$$\begin{aligned} &\Pi = H(\sigma, \theta) \\ &p^\top \Pi = p^\top, \quad \sum_{x \in Z, d \in \{0,1\}} p_{x,d} = 1 \end{aligned}$$

Once we have $p$, we can them compute the moments. For example,

$$\begin{aligned} m_x &= \sum_{x,d} p_{x,d}\, x, \;\; m_d = \sum_{x,d} p_{x,d}\, d \\ m_{xx} &= \sum_{x,d} p_{x,d}\, (x - m_x)^2, \;\; m_{xd} = \sum_{x,d} p_{x,d}\, (x - m_x)(d - m_d) \end{aligned}$$

and similarly for the other moments.

We are now ready to express the MPEC approach to moments estimation for the bus model. The objective function is the weighted squared differences between the data moments and the model moments

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

When we combine the objective and constraints, the optimization problem is

$$\begin{aligned} \max_{\sigma, \theta, \Pi, p, m} \quad & \mathcal{M}(m, M) \\ \text{s.t.} \quad & G(\sigma, \theta) = 0 \\ & \Pi = H(\sigma, \theta) \\ & p^\top \Pi = p^\top, \quad \sum_{x \in Z, d \in \{0,1\}} p_{x,d} = 1 \end{aligned}$$

$$m_x = \sum_{x,d} p_{x,d}\, x,\ m_d = \sum_{x,d} p_{x,d}\, d$$
$$m_{xx} = \sum_{x,d} p_{x,d}\, (x - m_x)^2,\ m_{xd} = \sum_{x,d} p_{x,d}\, (x - m_x)(d - m_d)$$
$$m_{dd} = \sum_{x,d} p_{x,d}\, (d - m_d)^2$$
$$m_{xxx} = \sum_{x,d} p_{x,d}\, (x - m_x)^3,\ m_{xxd} = \sum_{x,d} p_{x,d}\, (x - m_x)^2 (d - m_d)$$
$$m_{xdd} = \sum_{x,d} p_{x,d}\, (x - m_x)(d - m_d)^2,\ m_{ddd} = \sum_{x,d} p_{x,d}\, (d - m_d)^3$$

We applied this procedure to the same 20 data sets used in Table 3. The constrained method of moments problem was coded in AMPL, and submitted to the SNOPT solver via NEOS. Table 6 displays the results. Again we see that the method is fast. It is not as fast as maximum likelihood. That is not surprising given the larger size of the moments problem and the nonlinearity introduced by the constraints related to moments, particularly the skewness equations. We made no attempt to insure that the problem was properly scaled; poor scaling is almost automatically assured by the fact that some constraints use cubic powers of the state variable $x$ and some just used $x$.

Table 6: Method of Moments Parametric Bootstrap Results

| | Estimates | | | | | | CPU | Major | Evals | Bell. EQ |
|---|---|---|---|---|---|---|---|---|---|---|
| | $RC$ | $\theta_1^c$ | $\theta_2^c$ | $\theta_1^p$ | $\theta_2^p$ | $\theta_3^p$ | (sec) | Iter | | Error |
| **Mean** | 1.03 | 0.048 | 0.001 | 0.397 | 0.603 | 0.000 | 22.6 | 525 | 1753 | 6.7E−06 |
| **S.E.** | 0.31 | 0.032 | 0.002 | 0.040 | 0.040 | 0.001 | 16.9 | 389 | 1513 | 1.1E−05 |
| **Min** | 0.13 | 0.000 | 0.000 | 0.340 | 0.511 | 0.000 | 5.4 | 168 | 389 | 1.8E−10 |
| **Max** | 1.45 | 0.104 | 0.009 | 0.489 | 0.660 | 0.004 | 70.1 | 1823 | 6851 | 3.6E−05 |

Our example is a primitive approach to the method of moments. A more sophisticated implementation would repeat this with a weighting matrix implied by the initial solutions. Larger problems would have larger Markov transition matrices, but if we knew the sparseness structure then we could exploit it to efficiently represent the constraints that define the Markov transition matrix.

Simulated method of moments is often used in practice when it is difficult to compute the exact moments predicted by a parameterization of the model. To compute simulated moments, one could just define a procedure (subroutine) which takes a parameter vector, decision rules, and a fixed sequence of random draws and computes the implied moments.

Despite the simplicity of this example, it does show that moment estimators can be implemented within the MPEC framework.

## 8. Computational Notes

The examples we have used to exposit the key idea are all infinitesimal relative to the optimization problems that are commonly solved. The computers we used were all desktop computers with a single processor. These problems can easily take advantage of the multiple processor environments used in high-power and high-throughput computing environments.

The software we used is aimed at solving problems of moderate to large size. There are other methods that can be used to solve huge problems.

Our dynamic programming example discretized the state space. Such discretizations are impractical in many interesting problems. It is preferable to approximate value functions with linear combinations of basis functions, as is done in Judd (1992). In that case, the constraints become the projection conditions that solve the dynamic problem. The mathematical programming methods then become similar to a variety of problems currently solved in the engineering and science literatures, such as PDE constrained optimization.

Therefore, the small size and limited nature of our examples cannot be read as indications of size limits to the numerical optimization approach to solving structural estimation problems.

## 9. Conclusion

Maximum likelihood estimation, as well as other methods like methods of moments, indirect inference, partial identification, and calibration are often just constrained optimization problems. Standard theorems in mathematical programming argue that the nonlinear programming approach can be applied directly to those problems, contradicting the claims that numerical analysis does not offer economists any useful tools. Our examples show that this theoretical superiority appears in many familiar econometric contexts. This is valuable for estimation purposes since many of the "computationally light" alternatives, such as two-stage methods, are not asymptotically efficient. The fact that we used only the most basic numerical software and hardware available today, it is clear that the MPEC approach cannot be dismissed and could be very useful in many contexts.

## References

[1] Agguiregabiria, Victor, and Pedro Mira. "Sequential Estimation of Dynamic Discrete Games," Econometrica, 75(1) (January, 2007), 1-53.

[2] Aitchison, J. & S.D. Silvey. Maximum likelihood estimation of parameters subject to restraints. Annals of Mathematica1 Statistics 29 (1958): 813-828.

[3] Erdem, Tülin, Kannan Srinivasan, Wilfred Amaldoss, Patrick Bajari, Hai Che, Teck Ho, Wes Hutchinson, Michael Katz, Michael Keane, Robert Meyer, and Peter Reiss, "Theory-Driven Choice Models" (2004).

[4] Gallant, A. Ronald, and Alberto Holly. "Stastical Inference in an Implicit, Nonlinear, Simultaneous Equation Model in the Context of Maximum Likelihood Estimation" Econometrica, Vol. 48, No. 3 (April, 1980)

[5] Rust, John, Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher, Econometrica, 55 (1987) 999–1033.

[6] Rust, John, Nested Fixed Point Algorithm Documentation Manual: Version 6 (2000).

[7] Silvey, S. D. (1970), Statistical Inference, London: Chapman & Hall.

[8] 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

[9] Wolak, F. A. (1987). An exact test for multiple inequality and equality constraints in the linear regression model. J. Am. Statist. Assoc. 82, 782-93.

[10] Wolak, F.A. Testing inequality constraints in linear econometric models. Journal of Econometrics, 1989.

[11] Burguete, J.F., A.R. Gallant, and G. Souza. On unification of the asymptotic theory of nonlinear econometric models. Econometric Reviews 1 (1982): 151-190.