

A Framework for Indirect Inference

Paul L. Fackler and Huseyin Tastan*

June 5, 2008

Abstract

A general framework for simulation based indirect inference methodology is proposed. The framework highlights the distinction between the structural (economic) model that describes the data generating process and the reduced form (statistical) model used for estimation. The framework is illustrated with two simple examples. A MATLAB implementation is described and illustrated with worked examples.

Keywords: Simulated Method of Moments, Indirect Inference, Efficient Method of Moments

*The authors are Associate Professor at North Carolina State University and Assistant Professor at Department of Economics, Yildiz Technical University.

Mail: Paul L. Fackler

Department of Agricultural and Resource Economics

NCSU, Box 8109

Raleigh NC, 27695, USA

e-mail: paul_fackler@ncsu.edu

Web-site: <http://www4.ncsu.edu/~pfackler/>

© 2003-4, Paul L. Fackler

1 Introduction

Indirect inference estimation addresses a long standing problem in econometrics. Analysts often specify a model that relates parameters and exogenous variables of an economic model to some set of observable (endogenous) variables. In many situations the economic model is too complicated to admit useful expressions for the probability distributions associated with the endogenous variables. Even expressions for expectations of functions of the data may not exist. In such cases, fully efficient estimation procedures, such as maximum likelihood, may not be applicable.

It may be possible, however, to obtain simulated values of model variables cheaply using a typical personal computer for given (structural) model parameter values. Simulated generalized method of moments (GMM) methods estimate model parameters by matching properties of the simulated values to those in observed data. A number of simulation based estimation methods have been proposed to utilize such an approach. These methods are known by various names, including Simulated Method of Moments (SMM) (McFadden (1989), Pakes and Pollard (1989), Duffie and Singleton (1993) and Lee and Ingram (1991)), Indirect Inference (II) (Smith (1993), Gouriéroux, Monfort, and Renault (1993) and Gouriéroux and Monfort (1996)) and Efficient Method of Moments (EMM) (Gallant and Tauchen (1996)).

Applications of simulation-based estimation methods have appeared in diverse areas in economics. Among these are stochastic volatility and equity return models (Durham (2006), Andersen, Benzoni, and Lund (2002), Liu (2000), Andersen, Chung, and Sorensen (1999), Monfardini (1998), Andersen and Lund (1997), van der Sluis (1997), Gallant, Hsieh, and Tauchen (1997)), exchange rate models (Bansal, Gallant, Hussey, and Tauchen (1995), Chung and Tauchen (2001)), commodity price and storage models (Michaelides and Ng (2000)), ARFIMA and VARFIMA models (Martin and Wilkins (1999)), ARMA models (DeLuna and Genton (2001); Chumacero (1997), Chumacero (2001), Ghysels, Khalaf, and Vodonou (1994)), SDE models (Cleur and Manfredi (1999), Long (1997)), real business cycle models (Smith (1993)) and discrete response models (McFadden (1989)). Ghysels and Guay (2003) and Ghysels and Guay (2004) proposed a set of structural change tests for models estimated using simulated method of moments.

Econometricians desiring to use simulated GMM methods are hampered by the lack of standardized notation and software. The literature contains a number of approaches for which the essential similarities and differences are masked by different names and notation for the same concepts. Furthermore, the approach has not been incorporated into standard econometric packages. In part this is due to the lack of a common framework and in part due to the requirement that the package incorporates a flexible modeling language.

This paper addresses both shortcomings. It provides a common notation that links the various simulated GMM methods that have appeared in the literature.

In doing so it highlights the computational issues in obtaining estimates and make possible the development of a simple set of estimation tools. It also describes a set of functions, written in the MATLAB programming language, that allow an analyst to concentrate on model and moment specification rather than the details of computational algorithms. The code is freely available for non-commercial users from the first author's web site.

The next section develops a common notation to describe indirect inference and discussed how previously proposed estimators can be described in terms of this notation. This is followed by two examples and then by a section describing use of the MATLAB functions that implement the approach.

2 Simulated GMM

The general purpose of simulation based estimation methods is to estimate the parameters of a structural model by simulating data from that model and determining the parameters that minimize a GMM type criterion function for some set of moment conditions. Simulated GMM models, therefore, consist of two distinct parts. The first part is a structural economic model that describes the mapping from parameters, shocks and exogenous variables to endogenous variables. The second part is a set of moment conditions used for estimation.

To make the framework specific, let θ be a p -vector of structural parameters to be estimated. For simplicity, suppose that there are no exogenous variables,¹ so the inputs to the structural model consist only of the parameters and a set of shocks e with known distribution. The structural model produces a set of endogenous variables, represented by y : $y = y(\theta, e)$.

Simulation based estimation methods are most useful when direct inference about θ is difficult. Instead inference is performed using a set of q moment conditions $m(\beta, y)$, where $\beta \in R^q$ (with $q \geq p$) is a set of auxiliary (reduced form) parameters such that $E[m(\beta, y)] = 0$. Estimates of β are obtained by determining the value that makes sample averages of $m(\beta, y)$ close to zero. The basic idea of simulated GMM is to find a value of θ which yields similar value of β for actual data and for data simulated using $y(\theta, e)$.

Before continuing, it is useful to be more explicit about the size of the various components of the model. Suppose that T observations on y are available for estimation, stored in a T -row matrix \hat{y} . Also suppose that given a T vector of shocks and model parameters θ , one can simulate H realizations of endogenous variable from the structural model. The function $\tilde{y}(\theta, e)$ takes the p -vector θ and a TH -row matrix of shocks as inputs and returns a TH -row matrix \tilde{y} of simulated values of the endogenous variables.

The function $m(\theta, y)$ accepts a q -vector β and a T or TH -row matrix y (either

¹Details for handling exogenous variables are discussed in the section documenting the MATLAB implementation.

\hat{y} or \tilde{y}) and returns an $N \times q$ or $NH \times q$ matrix of observations on the q moment conditions, where $N \leq T$ (N may be less than T if some observations are used as initial conditions in models with serial dependence). Likewise the function $\beta(y)$ accepts a T -row or TH -row matrix y and returns a q -vector such that each column of $m(\beta(y), y)$ sums to zero.

There are three approaches to simulation based inference that are distinguished by how m is used. In each case a q -valued function $\psi(\theta)$ is defined and an estimator of θ is found by solving

$$\min_{\theta} \psi(\theta)^\top W \psi(\theta), \quad (1)$$

where W is some appropriately defined $q \times q$ weighting matrix.

Simulation of the endogenous variables is conducted using TH realizations of the shocks, denoted e_i , $i = 1, \dots, H$, where each e_i contains T observations. The observation of simulated data associated with e_i is denoted \tilde{y}_i . A value of β based on simulated data is denoted $\tilde{\beta} = \frac{1}{H} \sum_i \beta(y(\theta, e_i))$. The value of β based on the observed data is denoted $\hat{\beta} = \beta(\hat{y})$.

The first approach evaluates the moment function using the data based estimate of β and simulated values of y :

$$\psi_1(\theta) = \frac{1}{HN} \sum_i \sum_t m_t(\hat{\beta}, \tilde{y}_i), \quad (2)$$

where m_t denotes row t of $[m(\beta, y)]$, the t th observation of the moment function. The second approach uses values of β based on simulated data, together with the sample data:

$$\psi_2(\theta) = \frac{1}{N} \sum_t [m_t(\tilde{\beta}, \hat{y})]. \quad (3)$$

The third approach attempts to simply match values of β :²

$$\psi_3(\theta) = \tilde{\beta} - \hat{\beta}. \quad (4)$$

The three estimators are closely related and are asymptotically equivalent. In the exactly identified case ($p = q$), it is clear that when $\tilde{\beta} = \hat{\beta}$, $\sum_t m_t(\tilde{\beta}, \hat{y}) =$

²The third approach can be viewed as a first order approximation to the other approaches and is particularly close to the second method. To see this expand $m(\tilde{\beta}, \hat{y})$ around $\beta = \hat{\beta}$:

$$m(\tilde{\beta}, \hat{y}) = m(\hat{\beta}, \hat{y}) + \frac{\partial m(\hat{\beta}, \hat{y})}{\partial \beta} (\tilde{\beta} - \hat{\beta}) + h.o.t.$$

By definition $m(\hat{\beta}, \hat{y}) = 0$; hence

$$m(\tilde{\beta}, \hat{y}) \approx \frac{\partial m(\hat{\beta}, \hat{y})}{\partial \beta} (\tilde{\beta} - \hat{\beta})$$

$\frac{\partial m(\hat{\beta}, \hat{y})}{\partial \beta}$ is constant in θ , and therefore, to a first order approximation, $E[m(\tilde{\beta}, \hat{y})] = 0$ is equivalent to $E[\tilde{\beta} - \hat{\beta}] = 0$.

$\sum_t m_t(\tilde{\beta}, \hat{y}) = 0$. Thus, in exactly identified situations, the three methods yield the same estimator. The three approaches are also identical when m is additively separable:

$$m_t(\beta, y) = \beta - \mu_t(y) \quad (5)$$

This occurs, for example, when the reduced form parameters are sample moments of the data.

The choice of which estimator to use depends mainly on ease of estimation. There is little difference in this regard so long as $\beta(y)$ is easy to evaluate. The second method requires the most computation because it must update $\tilde{\beta}$ for each trial θ and then compute $m(\tilde{\beta}, \hat{y})$; it would therefore not typically be used in practice. In the additively separable case, the third method requires the least amount of computational work. When $\beta(y)$ is computationally intensive to evaluate, however, the first method has clear advantages, because $\hat{\beta}$ is evaluated once and then $m(\hat{\beta}, \tilde{y})$ is used.

Notice that the auxiliary model parameter vector β is defined implicitly in terms of the structural parameter vector θ :

$$b(\theta) = E_e[\beta(y(\theta, e))] \quad (6)$$

This mapping is called the binding function in (Gourieroux, Monfort, and Renault 1993) and (Gourieroux and Monfort 1996). If the binding function is known in closed form (as it sometimes is) simulation can be dispensed with and estimation conducted directly using $b(\theta) - \hat{\beta}$.

Currently there are three major existing approaches to simulation based GMM in the literature. The first, typically known as Simulated Method of Moments (SMM) (Duffie and Singleton (1993) and Lee and Ingram (1991)), uses a simple moment matching scheme. In the context of the current notation, this is the additively separable case in which $m(\beta, y) = \beta - \mu(y)$. In the literature the moment condition is typically described as $\mu(\tilde{y}) - \mu(\hat{y})$, which inspection reveals is identical to either $m(\tilde{\beta}, \hat{y})$ or $m(\hat{\beta}, \tilde{y})$. A second approach, generally called Indirect Inference (II) (Smith (1993), Gourieroux, Monfort, and Renault (1993) and Gourieroux and Monfort (1996)) describes the moment condition in terms of an auxiliary model that involves a parameter vector β , which is the solution to an optimization problem of the form

$$\max_{\beta} \sum_t Q_t(\beta, y)$$

In terms of the current notation

$$m_t(\beta, y) = \partial Q_t(\beta, y) / \partial \beta.$$

One can also use the moment condition given by

$$m(\beta, y) = \hat{\beta}(\hat{y}) - \tilde{\beta}(\tilde{y}(\theta, e))$$

where

$$\hat{\beta} = \operatorname{argmax} \sum_t^T Q_t(\beta, \hat{y}),$$

and

$$\tilde{\beta} = \operatorname{argmax} \sum_t^{TH} Q_t(\beta, \tilde{y}(\theta, \epsilon)).$$

Discussions of Indirect Inference generally use ψ_3 ($\tilde{\beta} - \hat{\beta}$) but there is no reason why the other forms of ψ cannot be used. As pointed out earlier, if it is difficult to find β , then ψ_1 ($m(\hat{\beta}, \tilde{y})$) will generally be preferred.

The third approach in the literature has been called Efficient Method of Moments (EMM) ((Gallant and Tauchen 1996)). This approach is also described in terms of an auxiliary model with a known score function $S_t(\beta, y)$. In the current notation, this score function is identical to the moment function $m_t(\beta, y)$. Although EMM is generally linked with ψ_1 ($m(\hat{\beta}, \tilde{y})$), as with II any of the three objective functions could be used.

EMM draws its name from the fact that if the auxiliary model encompasses the structural model, the resulting estimator is asymptotically efficient. Furthermore, if the likelihood function associated with the score function is a good approximation to the density implied by the structural model, EMM will be nearly efficient.³

Gallant and Tauchen (1996) suggest the use of ψ_1 ($m(\hat{\beta}, \tilde{y})$) as this avoids the need to recalculate β with simulated data. As previously mentioned, this will often be computationally advantageous when β requires numerical solution. It should not be thought, however, that EMM is superior to II. In fact EMM is a special case of II and the current notation makes clear that EMM and II are equivalent methods so long as the objective function used in the II procedure is a likelihood function for the auxiliary model, regardless of how ψ is defined.

To complete our discussion of Simulated GMM estimators, we provide a brief discussion of the optimal weighting matrix and asymptotic distribution theory. We assume that the usual regularity conditions are satisfied. For the ψ_1 and ψ_2 the optimal weighting matrix is $W = [\operatorname{Cov}(m_t(\beta, y))]^{-1}$ which in practice is approximated from the data. Let

$$\Gamma_j = \frac{1}{N} \sum_{t=j+1}^N m_t(\hat{\beta}, \hat{y}) m_{t-j}(\hat{\beta}, \hat{y})^\top \quad (7)$$

Standard practice uses the Newey-West approximation (Newey and West 1987):

$$\operatorname{Cov}(m_t(\beta, y)) = \Gamma_0 + \sum_{j=1}^J w_j (\Gamma_j + \Gamma_j^\top) \quad (8)$$

³For this reason, (Gallant and Tauchen 1996) suggest the use of a semi-nonparametric model to define the score, as such a model can be shown to approximate (and converge asymptotically to) a wide class of densities.

where J is referred to as the bandwidth and the w_j are weights. Common choices of weights are Bartlett $w_j = 1 - j/(J + 1)$ and Parzen⁴:

$$w_j = \begin{cases} 1 - 6x^2 + 6|x|^3, & \text{if } 0 \leq |x| \leq 0.5; \\ 2(1 - |x|)^3, & \text{if } 0.5 \leq |x| \leq 1, \end{cases}$$

where $x = j/(J + 1)$. If the moments can be safely assumed to be independent the bandwidth can be set to 0. For ψ_3 the optimal weighting matrix must be adjusted to account for the transformation from m to β :

$$W = \left[\frac{1}{N} \sum_t \frac{\partial m_t(\hat{\beta}, \hat{y})}{\partial \beta} \right]^\top \left[\Gamma_0 + \sum_{j=1}^J w_j (\Gamma_j + \Gamma_j^\top) \right]^{-1} \left[\frac{1}{N} \sum_t \frac{\partial m_t(\hat{\beta}, \hat{y})}{\partial \beta} \right] \quad (9)$$

To obtain an estimate of the asymptotic covariance of the SGMM estimator $\hat{\theta}$, let

$$D = \frac{1}{NH} \sum_{i=1}^H \sum_{t=1}^N \frac{\partial m_t(\hat{\beta}, y(\theta, e_i))}{\partial \theta} \quad (10)$$

Under the regularity conditions described in respective sources we have

$$\sqrt{N}(\hat{\theta} - \theta_0) \xrightarrow{d} N(0, \Omega),$$

where the covariance matrix can be approximated using

$$\hat{\Omega} = \left(1 + \frac{1}{H} \right) \left(D^\top \left[\Gamma_0 + \sum_{j=1}^J w_j (\Gamma_j + \Gamma_j^\top) \right]^{-1} D \right)^{-1}. \quad (11)$$

The $1 + 1/H$ term represents an adjustment for the errors introduced due to the simulation process. Asymptotic standard errors of the parameters can be calculated by

$$\text{ASE}(\hat{\theta}_i) = \sqrt{\frac{1}{N} \hat{\Omega}_{ii}}, \quad i = 1, \dots, p$$

where $\hat{\Omega}_{ii}$ is the i -th diagonal element.

3 Two Examples

The framework is first illustrated with a simple example for which an explicit solution to the mapping from θ to β exists. This, of course, obviates the need for the use of a simulation based approach. The example, however, serves to illustrate the method and make clear some of the issues involved in its use.

⁴For a review of alternative specifications for Long Run Covariance (HAC) matrix estimation in GMM framework see Hall (2005) pp.74-88.

Consider a simple demand and supply model in which the quantity demanded depends on the price of the good, an observable exogenous demand shifter and an unobserved shock. Similarly, the quantity supplied depends on the price of the good, an observable exogenous supply shifter and an unobserved shock, with the two unobserved shocks being independent. Specifically, the demand function is

$$q = a_d - b_d p + c_d x_1 + \sigma_d e_d, \quad (12)$$

and the supply function is

$$q = a_s + b_s p + c_s x_2 + \sigma_s e_s, \quad (13)$$

where $e \sim N(0, I_2)$. In the generic notation, $\theta = [a_d \ b_d \ c_d \ \sigma_d \ a_s \ b_s \ c_s \ \sigma_s]^\top$ and $y = [p \ q]$.

The model can be solved for p and q :

$$p = \frac{1}{b_s + b_d} (a_d - a_s + c_d x_1 - c_s x_2 + \sigma_d e_d - \sigma_s e_s) \quad (14)$$

and

$$q = \frac{1}{b_s + b_d} (b_s a_d + b_d a_s + b_s c_d x_1 + b_d c_s x_2 + b_s \sigma_d e_d + b_d \sigma_s e_s) \quad (15)$$

A simple set of moment conditions uses the additively separable approach

$$m(\beta, y) = \beta - [p \ q \ p^2 \ pq \ q^2 \ px_1 \ px_2 \ qx_1 \ qx_2]^\top \quad (16)$$

The moment conditions imply one overidentifying condition (if the demand and supply shocks were not independent, the model would be exactly identified).

As mentioned, an explicit mapping from θ to β (the so-called binding function) is easily derived in this case:

$$\begin{aligned} \beta_1 &= \frac{1}{b_s + b_d} (a_d - a_s + c_d \overline{x_1} - c_s \overline{x_2}) \\ \beta_2 &= \frac{1}{b_s + b_d} (b_s a_d + b_d a_s + b_s c_d \overline{x_1} + b_d c_s \overline{x_2}) \\ \beta_3 &= \frac{1}{b_s + b_d} (\sigma_d^2 + \sigma_s^2) + \beta_1^2 \\ \beta_4 &= \frac{1}{b_s + b_d} (b_s \sigma_d^2 - b_d \sigma_s^2) + \beta_1 \beta_2 \\ \beta_5 &= \frac{1}{b_s + b_d} (b_s^2 \sigma_d^2 + b_d^2 \sigma_s^2) + \beta_2^2 \\ \beta_6 &= \frac{1}{b_s + b_d} \left((a_d - a_s) \overline{x_1} + c_d \overline{x_1^2} - c_s \overline{x_1 x_2} \right) \\ \beta_7 &= \frac{1}{b_s + b_d} \left((a_d - a_s) \overline{x_2} + c_d \overline{x_1 x_2} - c_s \overline{x_2^2} \right) \\ \beta_8 &= \frac{1}{b_s + b_d} \left((b_s a_d + b_d a_s) \overline{x_1} + b_s c_d \overline{x_1^2} + b_d c_s \overline{x_1 x_2} \right) \\ \beta_9 &= \frac{1}{b_s + b_d} \left((b_s a_d + b_d a_s) \overline{x_2} + b_s c_d \overline{x_1 x_2} + b_d c_s \overline{x_2^2} \right) \end{aligned} \quad (17)$$

where the overline indicates averages of functions of the exogenous variables.

A second example involves an economic model for which no other simple methodology for estimation exists. It concerns a situation in which prices of a homogeneous good are observed at different spatial locations. The economic model

assumes that the excess demand for each location in each period can be well represented by a linear function

$$q_{it} = b_i(a_{it} - p_{it}), \quad (18)$$

where q_{it} and p_{it} represent the net imports and the price at location i in period t , b_i is a slope parameter and a_{it} is the exogenous autarky price for location i in period t . In addition, there are transport costs to move the good between each pair of locations. Let r_{ijt} be the cost of moving the good from location i to location j in period t .

The equilibrium conditions for this model are most easily expressed in terms of the shipment flows. Let s_{ijt} be the (non-negative) amount shipped from location i to j in period t . The net excess demand in location i is

$$q_{it} = \sum_k (s_{ikt} - s_{kit}) \quad (19)$$

The location i price is, therefore,

$$p_{it} = a_{it} - \frac{1}{b_i} \sum_k (s_{ikt} - s_{kit}) \quad (20)$$

In equilibrium, the price differential between any two locations cannot exceed the transport cost:

$$p_{it} - p_{jt} + r_{ijt} \geq 0 \quad (21)$$

Furthermore, if $s_{ijt} > 0$ this condition must be satisfied with equality. The equilibrium shipment amounts, therefore, solve the linear complementarity problem

$$a_{it} - \frac{1}{b_i} \sum_k (s_{ikt} - s_{kit}) - a_{jt} + \frac{1}{b_j} \sum_k (s_{jkt} - s_{kjt}) + r_{ijt} \geq 0$$

$$s_{ijt} \geq 0$$

with one of these two conditions satisfied with equality, for all i and $j \neq i$. Once the equilibrium shipments are found, the equilibrium prices can be computed.

The autarky prices and transport rates are considered to be latent variables which will be collectively denoted as z_t . It is assumed that z are normally distributed

$$z_t = \mu + Le_t$$

where e is i.i.d. $N(0, I)$, μ is an n -vector and L is an $n \times n$ lower triangular matrix.

In the economic simulation model the structural parameter vector θ is composed of the parameters μ , L and the b_i . For a given realization of e , sample values of autarky prices and transport rates can be generated and, for each realization of z_t the equilibrium prices are then computed.

Sample moments could be used as the auxiliary model. A better approach, however, fits an semi-nonparametric (SNP) density to the price data using model a selection criterion such as Akaike's Information Criterion, to choose the appropriate expansion order. The structural model can then be fit using the score function of the SNP density as the the moment conditions.⁵

4 A MATLAB Toolbox

Generic software to implement SGMM can be designed in such a way that an econometrician would need to write two functions and pass these functions along with data and command parameters to a solver. The first of the two functions defines the economic model by computing simulated data. When passed parameter values and shocks, it returns simulated endogenous variables $y(\theta, e)$. The second function defines the moment conditions used for estimation. When passed β and y , this function returns $m(\beta, y)$. Ideally this function also would return β if passed only y ($\beta = m(y)$).

This section describes a MATLAB implementation of the SGMM procedure. It relies on a number of functions available in the CompEcon Toolbox described in Miranda and Fackler (2002) and available at Fackler's website:

`www4.ncsu.edu/~pfackler/compecon`

The analyst must write two functions, which are passed, along with data and shocks, to the solver. The first function is the structural simulation model and is called using the syntax

```
y=simfunc(theta,e,SP1,SP2,...);
```

where SP1, SP2, etc., are optional additional parameters. These might be, for example, values of exogenous observable variables, control parameters guiding the simulation or parameters defining constraints on θ .

For the demand/supply model discussed in the previous section the function could be coded as follows:

```
function y=dssimfunc(theta,e,x);
ad      = theta(1);
bd      = theta(2);
cd      = theta(3);
sigd    = theta(4);
as      = theta(5);
```

⁵There are a number of identification issues involved with this model. For example, if a particular transport link is never used, the parameters associated with the transport costs on this link are not identified. Furthermore, if the trade patterns among locations are very stable, it may not be possible to identify all of the demand related parameters. A more complete discussion of the identification issues for this model is available from the authors.

```

bs      = theta(6);
cs      = theta(7);
sigs    = theta(8);
p = (ad-as+cd*x(:,1)-cs*x(:,2)+sigd*e(:,1)-sigs*e(:,2))/(bs+bd);
q = as+bs*p+cs*x(:,2)+sigs*e(:,2);
y = [p q];

```

The second function that must be coded by the analyst is the moment function. The syntax for this function has two forms:

```
m=momfunc(beta,y,avg,MP1,MP2,...);
```

or

```
beta=momfunc([],y,avg,MP1,MP2,...);
```

The `avg` input is a 0/1 variable. If 1, the moment conditions should be averaged over all observations, if 0, moment conditions should be returned for each observation. The inputs `MP1`, `MP2`, etc. are optional additional parameters passed to the simulation function. The first form returns a $q \times 1$ vector (if `avg=1`) or an $N \times q$ matrix of N observations on q moment conditions (if `avg=0`). With the second syntax, the `beta` parameter is passed as an empty matrix and the function should return the optimal value of β (with this syntax `avg` is ignored). It must be the case that

```
sum(momfunc(momfunc([],y),y,0))
```

is within rounding error of a $1 \times q$ vector of zeros. If an empty matrix is returned using the second syntax, the toolbox solver will call a routine to determine the appropriate value of β .

It is not essential that the moment function be able to solve for β . With the EMM framework, β is solved only once with the actual data. This value of β may then be passed to the SGMM solver. This facilitates situations in which the optimal β may be difficult or require some experimentation to find. For example, if the SNP score function is used for moment conditions, the determination of the expansion order may involve estimation of several orders and the use of a model selection criterion to make a final choice.

The moment function for the demand/supply example could be coded

```

function out=dsmomfunc(beta,y,avg,x);
m=[crossmom(y,2) dprod(y,x)];
if isempty(beta)
    out=mean(m)';
elseif avg==1
    out=sum(m)'/size(m,1)-beta;
else
    beta=beta';

```

```

    out=m-beta(ones(size(m,1),1),:);
end

```

The function `crossmom` computes the products and crossproducts up to order k . In this case the i th row produced by `crossmom(y,2)`, where y has two columns, is

$$[y_{1i} \quad y_{2i} \quad y_{1i}^2 \quad y_{1i}y_{2i} \quad y_{2i}^2]$$

The `dprod` (direct product) function computes the row-wise Kronecker product; in this case the i th row is

$$[y_{1i}x_{1i} \quad y_{1i}x_{2i} \quad y_{2i}x_{1i} \quad y_{2i}x_{2i}]$$

Two “generic” moment functions are described in an appendix. They are useful on their own and also provide further examples of how to code moment functions. The two functions return the moments and cross moments up to order k and the score function associated with the multivariate semi-nonparametric (SNP) density function (Gallant and Nychka (1987), Gallant and Tauchen (1989)). In the latter case, the procedure does not compute β when passed an empty matrix because this is a non-trivial optimization problem that should be performed prior to calling the SGMM solver.

The solver is called using the syntax

```
[theta,objval,V,dpsi]=gmmsim(theta0,simfunc,e,sp,beta0,momfunc,y,mp,exog,W);
```

The input variables are:

theta0 : a $p \times 1$ vector of initial values of the structural parameters
simfunc : a string variable naming the user defined simulation function
e : a 2 or 3-dimensional array of shocks with second dimension of size H
sp : a cell array of optional parameters for the simulation function:
: **sp**={SP1,SP2,...}
beta0 : a $q \times 1$ parameter vector of the auxiliary model
momfunc : a string variable naming the user defined moment function
y : a matrix of endogenous data values
mp : a cell array of optional parameters for the moment function:
: **mp**={MP1,MP2,...}
exog : optional argument, locations of exogenous variables in **sp** and **mp**
W : user-supplied weighting matrix, optional.

The output variables are:

theta : a $p \times 1$ vector of the structural parameter estimates
objval : value of objective function at parameter estimates
V : covariance matrix of parameter estimates
dpsi : gradient of moments

The optional parameters **sp** and **mp** should be passed as empty cell arrays (`{ }`) if the associated functions do not use additional parameters. **beta0** is only needed if $\hat{\beta}$ is computed by the solver (i.e., if **momfunc** returns an empty value of **beta** when passed an empty value of **beta**); in other situations, **beta0** is not used and can be passed to the solver as an empty variable. The solver will calculate optimal weighting matrix when **W** is not supplied by the user or when **W**=`[]`.

An important issue in simulated GMM is how to handle exogenous observed data. Suppose we have T observations on the endogenous variable y and on the exogenous variable x . It is typically desirable, however, to use far more simulated observations in computing $\psi(\theta)$. If the number of shocks used for simulation is an even multiple of T (i.e., if H is an integer), the exogenous variables can be “reused” H times in generating the simulated values.

There are two ways to do this. One is to do it within the simulation and moment functions. This can be accomplished by adding to the simulation file the line

```
if size(e,1)>size(x,1), x=repmat(x,size(e,1)/size(x,1),1); end
```

adding to the moment file the line

```
if size(y,1)>size(x,1), x=repmat(x,size(y,1)/size(x,1),1); end
```

(both lines assume there is only one exogenous variable called **x**). For some problems, it may be necessary to loop over the H shock samples rather than simply replicating the exogenous data. For example, if there are initial conditions that must be handled in a dynamic model, it may be more straightforward to use a loop than to use data replication.

The other approach to handling exogenous variables is to have the solver perform the replication by using the optional **exog** variable. **exog** is a list of locations in the cell arrays **sp** and **mp** representing exogenous variables. The exogenous data must be passed in the same locations in the two cell arrays. For example, if variables 1, 3 and 4 are exogenous variables, pass `[1 3 4]` as **exog** to the solver. This informs the solver that these variables must be expanded before being passed to the simulation and moment functions. If there are no exogenous variables, **exog** need not be passed or should be set to empty.

For the demand/supply model, given y and x (both $T \times 2$), the solver would be called using:

```
e=randn(T*H,2);
theta=gmmSim(theta0,'dssimfunc',e,{x},[],'dsmomfunc',y,{x},1);
```

The exogenous variables will be expanded within the solver as the last argument of **gmmSim** is set to 1 (**exog**). Alternatively, instead of passing one long realization of **e**, users may prefer performing replications within the simulation and moment functions. In this case the calling syntax will be:

```
e=randn(T,H,2);
theta=gmmssim(theta0,'dssimfunc',e,{x},[],'dsmomfunc',y,{x});
```

Note that `exog` argument is not passed to `gmmssim`. Notice also that, in this case, simulation and moment functions should be coded such that they can handle three-dimensional arrays. For the demand and supply model these are equivalent because the moment function is additively separable. If the moment function involves parameters of an auxiliary model that needs to be estimated conditionally on some initial conditions, then the replications over H should be performed within the simulation and moment functions.

As a practical matter, it may be useful to get a rough estimate of θ using a low value of H , as iteration speed is inversely related to simulation size. The estimates can then be refined with a larger value of H :

```
H=2; e=randn(T*H,2);
theta1=gmmssim(theta0,'dssimfunc',e,{x},[],'dsmomfunc',y,{x},1);
H=25; e=randn(T*H,2);
theta2=gmmssim(theta1,'dssimfunc',e,{x},[],'dsmomfunc',y,{x},1);
```

Optimization algorithms may also be unable to make progress toward a solution. Monitoring progress and restarting with new shocks may be useful in the small H case to attempt to rapidly find a near solution.

For the gradient based methods (quasi-Newton and Gauss-Newton) the gradient of the moments with respect to the structural parameters is required. It may be possible to derive this explicitly in the form

$$\frac{dm(\beta, y_t)}{d\theta} = \frac{dm(\beta, y_t)}{dy_t} \frac{dy_t}{\theta}$$

or in the form

$$\frac{d\beta(y_t)}{d\theta} = \frac{d\beta(y_t)}{dy_t} \frac{dy_t}{\theta}$$

The two generic moment functions are designed to provide a second output equal to either $dm(\beta, y_t)/dy_t$ or $d\beta/dy$ (both as $N \times q$ matrices). To use this feature, the structural simulation model must also be written to return $dy_t/d\theta$ (an $N \times p$ matrix). This feature is optional and if not utilized, the solver will use numerical derivatives.

The code is designed to solve the GMM optimization problem using one of three optimization algorithms, Nelder-Mead, Gauss-Newton or quasi-Newton (Judd (1998); Miranda and Fackler (2002)) (the default is quasi-Newton) and any of the three objective functions discussed above can be requested (the default uses ψ_1 unless the moment function is additively separable). The code calls a function `GMMW` to compute the optimal weighting matrix. This function uses the Newey-West procedure with Bartlett weights. The number of lags with non-zero weights can be specified by the user (the default is 0). The code also uses one- or two-sided finite

difference approximations for the derivatives of the moment function (unless these are provided explicitly as discussed above).

Various options can be set using the `optset` function to control program operation. For example, `optset('gmmsim','maxit',100)` sets the maximum number of iterations used by the optimization algorithm. The following table lists the available options:

<code>alg</code>	Defines which version of ψ is used: 1 - $m(\hat{\beta}, \tilde{y})$ [the default] 2 - $m(\tilde{\beta}, \hat{y})$ 3 - $\tilde{\beta} - \hat{\beta}$
<code>wlags</code>	# of lags used in computing weighting matrix [0]
<code>optmeth</code>	optimization algorithm used: 'n' Nelder-Mead 'g' Gauss-Newton 'q' Quasi-Newton [the default]
<code>tol</code>	convergence tolerance used by optimization algorithm [1e-5]
<code>fdtype</code>	1 or 2 for 1 or 2-sided numerical derivatives
<code>fdstep</code>	steplength parameter used for numerical derivatives [1e-4]
<code>showiters</code>	1 if summary information shown at each iteration
<code>maxit</code>	maximum number of iterations [250]

One further feature of `gmmsim` is the ability to perform ordinary GMM when the binding function $b(\theta)$ can be computed directly. In this case the binding function should be coded with the syntax:

```
beta=bfunc(theta,bf);
```

or

```
[beta,dbeta]=bfunc(theta,bf);
```

where `bf` is a cell array of optional parameters. The second syntax provides both $b(\theta)$ and $db(\theta)/d\theta$; if there is no second output provided `gmmsim` will use numerical derivatives when needed. The solver is called using the syntax:

```
theta=gmmsim(theta0,bfunc,[],bf,beta0,momfunc,y,mp);
```

The empty value of the `e` input signals that the binding function is available.

Other outputs than the estimated θ vector are also available from `gmmsim`.

```
[theta,objval,V,dpsi]=gmmsim(...);
```

The second output is the value of the objective function. This can be used to test overidentifying restrictions as the asymptotic distribution of the value of the objective function multiplied by the number of observations under the null hypothesis that the structural model is correct is χ_{q-p}^2 . The third output `V` is an estimate of the $p \times p$ asymptotic variance/covariance matrix of the estimate of θ .

Table 1. Estimation Results for Demand/Supply Model

Parameter	Actual	Binding	Simulated	ASE
		Function	GMM	
a_d	6.0000	5.9681	5.9739	0.8286
b_d	0.7500	0.7413	0.7448	0.2083
c_d	1.0000	0.9967	0.9991	0.8789
σ_d	0.2500	0.3236	0.2591	0.3483
a_s	3.0000	3.2242	3.2204	1.9410
b_s	0.7500	0.6374	0.6335	0.8399
c_s	1.0000	0.9965	1.0132	0.5508
σ_s	0.7500	0.6179	0.6622	0.4191

Notes: Results are from one simulation of size 500 from demand-supply model. Estimation results are based on algorithm 3. ASE stands for asymptotic standard error.

The fourth output `dpsi` is an estimate of $d\psi/d\theta$. This can be useful in diagnosing identification problems. A model is (locally) identified if the rank of this $q \times p$ matrix is equal to p .

To illustrate some results, 500 observations of simulated data for the demand/supply model was generated. The structural parameters were then estimated using the exact binding function and using simulated GMM. For a single “typical” run, the former method took 0.77 seconds and the simulated approach took 40.81 seconds to converge. The actual and estimated parameters values are displayed in Table 1.

It is evident that the simulated values do not necessarily equal the values estimated using the known binding function. There are two reasons for this. First, the simulation based estimator does not compute the “true” value of $\beta(\theta)$ (or, therefore, the objective function) due to the inherent sampling error arising from the simulation. To see this, we have simulated the structural model using the actual θ (see Table 1) and calculated moments for each sample. The extent of the sampling error introduced by simulation can be seen from Figure 1. Second, both estimates do not find the exact optimal values of θ because the numerical optimizer stops well a convergence criteria is met. Making this criteria too strict would accomplish little in practice. This is evident in Table 2, which displays the sample $\hat{\beta}$ and the values of β computed using the binding function. The objective function attempts to make these as nearly equal as possible.

A second illustrative example was also carried out using the three-location spatial price equilibrium model discussed previously.⁶ In this model, autarky prices (a_t) and transaction costs (r_t) are treated as latent variables and 500 observations of data were simulated from $z_t = \mu + Le_t$ where $z_t = [a_t \ r_t]^\top$, e is i.i.d. $N(0, 1)$,

⁶The primary objective of estimating this model is to construct measures of market integration using the parameter estimates (see the discussion in Fackler and Tastan (2008); Tastan (2003); Fackler and Goodwin (2001) and McNew and Fackler (1997)).

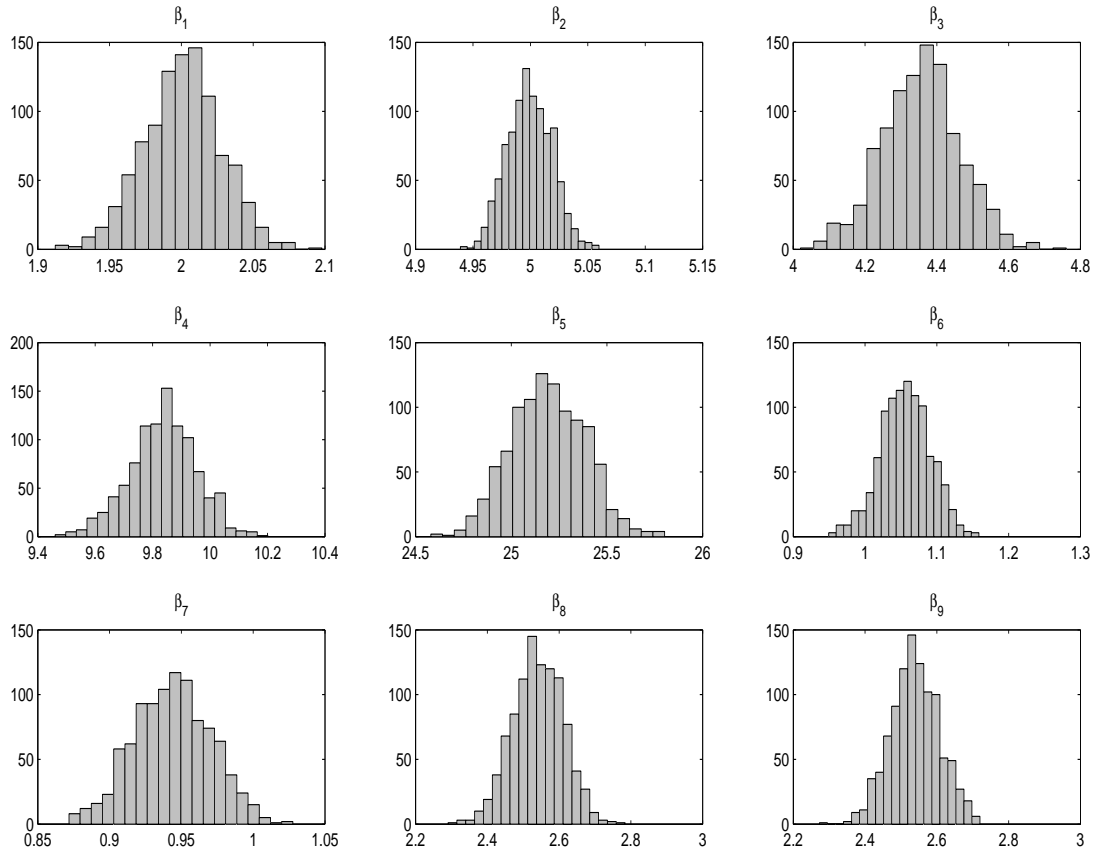


Figure 1: Mapping from actual θ to β

Table 2. Estimation Results for Demand/Supply Model

Sample	Actual	Binding Function	Simulated GMM
β_1	2.0217	2.0221	2.0219
β_2	5.0005	5.0002	5.0003
β_3	4.4197	4.4211	4.4206
β_4	9.9542	9.9553	9.9549
β_5	25.2042	25.2010	25.2020
β_6	1.0648	1.0649	1.0649
β_7	0.9445	0.9437	0.9439
β_8	2.5609	2.5605	2.5606
β_9	2.5500	2.5475	2.5483

μ is 5-vector and L is 5×5 lower triangular matrix. The underlying parameter vector is then $\theta = [\mu^\top \text{vec}(L)^\top \omega_1 \omega_2]^\top$. The following parameter values were used to simulate the model:

$$\mu = \begin{bmatrix} 10 \\ 10 \\ 12 \\ 1 \\ 1.2 \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 1.1 & & & & \\ 0.6 & 1 & & & \\ 0.8 & 0.9 & 1 & & \\ 0 & 0 & 0 & 0.25 & \\ 0 & 0 & 0 & 0.1 & 0.15 \end{bmatrix}$$

Notice that the underlying autarky prices and transaction costs are assumed to be independent of each other to reduce the number of parameters to be estimated. To further reduce the number of parameters we restrict the number of connections or trade linkages that are potentially active. We assume that there are only two potentially active links: location 1 ships to 3 and location 2 ships to 3. Location 3 is assumed to be a net importer at all times and trade linkages among locations switch among four possible regimes. In addition to partially observed underlying variables, the nonlinearity introduced by trade regime switches between time periods complicates the estimation problem. The restrictions on the potential trade linkages can be introduced by the following adjacency matrix:

$$Z = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix}$$

where -1 and 1 denote exporting and importing regions, respectively (in our example 1 ships to 3 and/or 2 ships to 3).

Similar to the previous demand/supply example the MATLAB implementation requires users to code two functions: a simulation function for the endogenous variables (prices) in the model and a moment function. The simulation function can be coded as follows

```
function [p,dp,S,q]=spatpricesim(theta,e,Z,mask,omfunc);
n=size(e,1);
[m,d]=size(Z);
mu=theta(1:d+m)';
L=zeros((d+m)*(d+m+1)/2,1); L(mask)=theta(d+m+1:end-d+1,1);
L=vechinv(L,2);
if omfunc
    [omega,domega]=r2simplex(theta(end-d+2:end));
else
    [omega,domega]=u2simplex(theta(end-d+2:end));
end
v=e*L'+mu(ones(n,1),:);

if nargout>3
```

```

        [p,S,q]=spatpricesolve(Z,v(:,1:d),v(:,d+1:end),omega);
    else
        [p,S]=spatpricesolve(Z,v(:,1:d),v(:,d+1:end),omega);
    end

```

This function calls the `spatpricesolve` function which solves the model using one of linear complementarity algorithms (such as Lemke's). In this example, we used `snpmom` function described in the Appendix with `k=3` for the order of polynomial expansion. The SNP order was chosen using Schwarz' Criterion. The results based on simulated data are shown in Table 3. We employed two numerical optimization routine to estimate this model. First, we used divided rectangles global optimization (DiRect) algorithm developed by Jones, Perttunen, and Stuckman (1993) to explore the surface of the objective function within initial parameter bounds. Then, the solutions produced by the DiRect algorithm are used as the starting values for the quasi-Newton algorithm.

The estimation results for the simulated data is given in Table 3. The results indicate that given the initial bounds on parameter space, the means of the underlying forcing variables are estimated reasonably well. Table 4 displays means, variances and correlation coefficients for actual data and simulated data based on the parameter estimates in Table 3. As can also be seen in Figure 2, which displays kernel density estimates of prices, simulation estimator is remarkably successful in matching the central tendency in prices. Variances of prices, however, are all underestimated. Figure 3 displays contours plots of bivariate price densities for both actual and simulated data. Although underlying forcing variables in the model are Gaussian, equilibrium prices are not. Simulation estimators are reasonably successful in catching elliptical shape of the bivariate price densities. The estimation results in Table 3 also indicate that some of the parameter estimates (for example L_9 , ω_1 and ω_2) did not converge to their true values. Simulation based estimation methods require global optima. If the optimization algorithm stops at a local minima then the simulation based estimators will not possess consistency and efficiency. These results make clear the need for a simulation study designed to assess properties of simulation based estimators within this model.

Table 3 Estimation Results for the Spatial Equilibrium Model

Parameter	Actual	Starting Values	DiRect	Qnewton	Std Errors
		[lower,upper]	Estimates	Estimates	
μ_{a_1}	10.0000	[8.0000, 12.0000]	10.0000	9.9987	0.0602
μ_{a_2}	10.0000	[7.0000, 13.0000]	10.0000	10.0015	0.0764
μ_{a_3}	12.0000	[8.0000, 15.0000]	13.8333	13.8335	2.1200
$\mu_{r_{13}}$	1.0000	[0.0500, 2.0000]	1.0714	1.0715	0.0227
$\mu_{r_{23}}$	1.2000	[0.0500, 2.0000]	1.0253	1.0245	0.0082
L_1	1.1000	[0.1000, 3.0000]	0.9069	0.9077	0.0413
L_2	0.6000	[-0.1000, 2.0000]	0.4734	0.4725	0.0494
L_3	0.8000	[-0.0100, 2.0000]	0.7992	0.7992	0.1880
L_4	1.0000	[0.0100, 2.0000]	1.0050	1.0032	0.0451
L_5	0.9000	[-0.0100, 1.0000]	0.5495	0.5493	0.3404
L_6	1.0000	[0.0010, 2.0000]	1.4447	1.4450	0.8153
L_7	0.2500	[0.0010, 1.0000]	0.1698	0.1698	0.0305
L_8	0.1000	[0.0010, 0.5000]	0.0837	0.0848	0.0399
L_9	0.1500	[0.0010, 0.5000]	0.3056	0.3043	0.0087
ω_1	0.2500	[0.0500, 0.9000]	0.5784	0.6865	0.1029
ω_2	0.5000	[0.0500, 0.9000]	0.0974	0.0740	0.0435

Table 4 Estimation Results for the Spatial Equilibrium Model

	Actual		Simulation	
	Mean	Variance	Mean	Variance
p_1	10.3442	1.1322	10.3107	0.7728
p_2	10.2268	1.2668	10.1946	1.0942
p_3	11.0821	1.3367	11.0462	1.0020

Price Correlations		
	Actual	Simulated
$\text{corr}(p_1, p_2)$	0.7855	0.7225
$\text{corr}(p_1, p_3)$	0.8768	0.8207
$\text{corr}(p_2, p_3)$	0.8961	0.8999

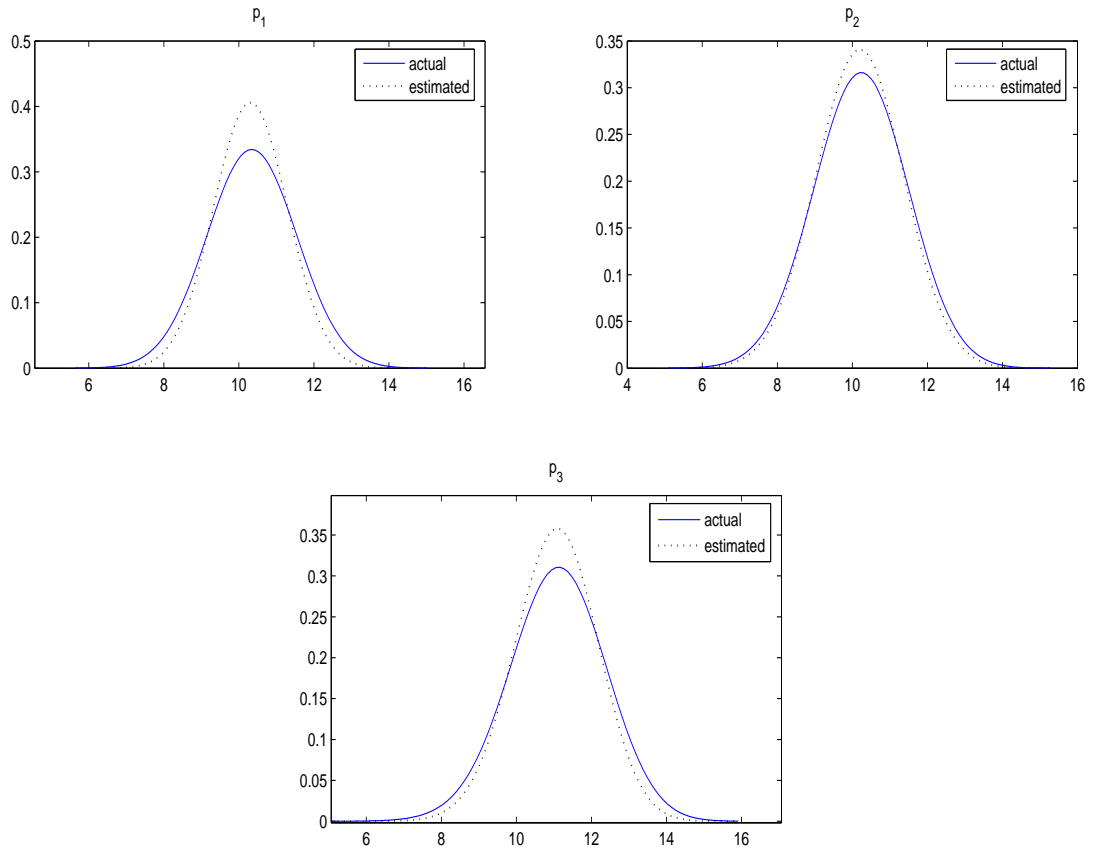


Figure 2: Kernel density estimates of prices. Solid: actual, dot: simulated

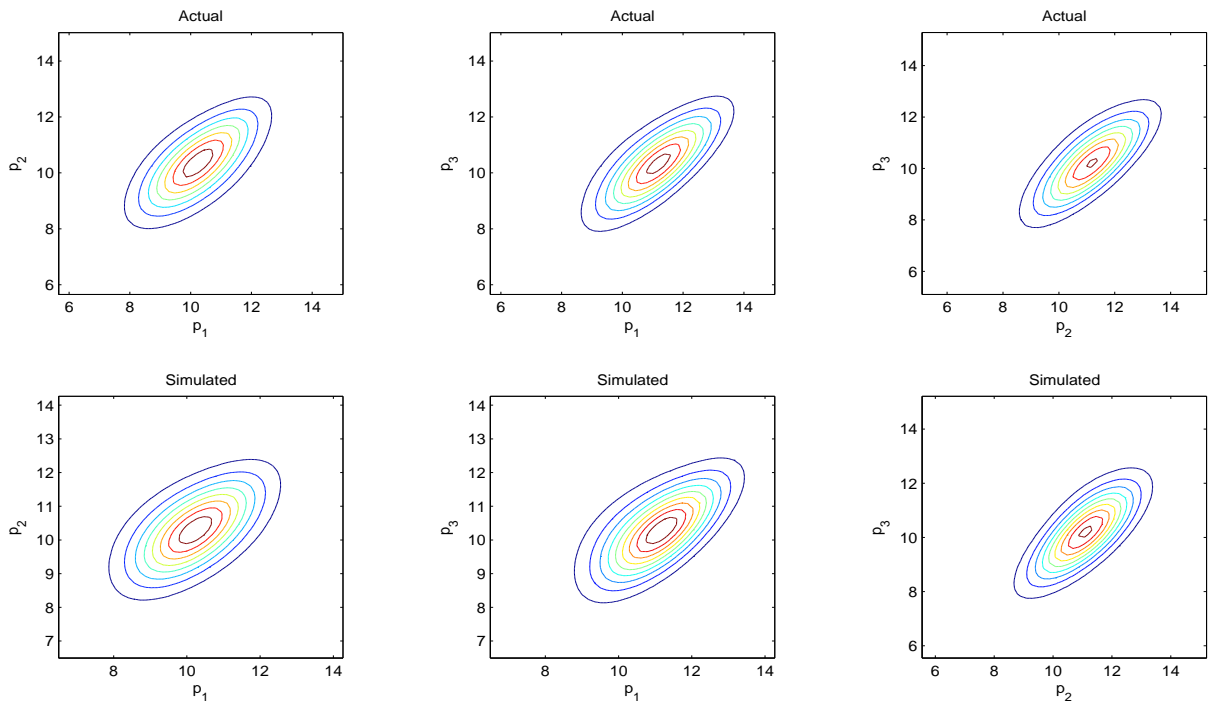


Figure 3: Contour plots of bivariate price densities. First row: actual, Second row: simulated

5 Summary

This paper has described a framework for using indirect inference to estimate parameters of structural models that may be too complicated to use more common econometric techniques. The method of indirect inference makes clear the distinction between the economic (structural) model and the statistical (reduced form) model. The parameters of the former, θ , are of main interest, but the parameters of the latter, β , are easier to estimate from data. Indirect inference attempts to estimate θ by matching the β obtained from simulated data as closely as possible to the β obtained from sample data.

The approach is illustrated with simple economic examples that clearly illustrate the technique and are accessible to anyone with elementary economics training. Also provided is documentation for a program, written in MATLAB, that will make this approach more accessible to practitioners. The program allows the analyst to concentrate more on model and moment specification and less on the details of solution algorithms.

References

- ANDERSEN, T., L. BENZONI, AND J. LUND (2002): “An Empirical Investigation of Continuous-Time Equity Return Models,” *Journal of Finance*, LVII(3), 1239–1284.
- ANDERSEN, T., H. CHUNG, AND B. SORENSEN (1999): “Efficient Method of Moments Estimation of a Stochastic Volatility Model,” *Journal of Econometrics*, 91, 61–87.
- ANDERSEN, T. G., AND J. LUND (1997): “Estimating Continuous-time Stochastic Volatility Models of the Short-term Interest Rate,” *Journal of Econometrics*, 77, 343–377.
- BANSAL, R., A. GALLANT, R. HUSSEY, AND R. TAUCHEN (1995): “Nonparametric Estimation of Structural Models for High-frequency Currency Market Data,” *Journal of Econometrics*, 66, 251–287.
- CHUMACERO, R. A. (1997): “Finite Sample Properties of the Efficient Method of Moments,” *Studies in Nonlinear Dynamics and Econometrics*, 2(2), 35–51.
- (2001): “Estimating ARMA Models Efficiently,” *Studies in Nonlinear Dynamics and Econometrics*, 5(2), 103–114.
- CHUNG, C., AND G. TAUCHEN (2001): “Testing Target-Zone Models Using Efficient Method of Moments,” *Journal of Business and Economic Statistics*, 19(3), 255–269.
- CLEUR, E., AND P. MANFREDI (1999): “One Dimensional SDE Models, Low Order Numerical Methods and Simulation Based Estimation: A Comparison of Alternative Estimators,” *Computational Economics*, 13, 177–197.
- DELUNA, X., AND M. GENTON (2001): “Robust Simulation-Based Estimation of ARMA Models,” *Journal of Computational and Graphical Statistics*, 10(2), 370–387.
- DUFFIE, D., AND K. SINGLETON (1993): “Simulated Moments Estimation of Markov Models of Asset Prices,” *Econometrica*, 61(4), 929–952.
- DURHAM, G. B. (2006): “Monte Carlo Methods for Estimating, Smoothing and Filtering One and Two-factor Stochastic Volatility Models,” *Journal of Econometrics*, 133, 273–305.
- FACKLER, P., AND B. GOODWIN (2001): “Spatial Price Analysis,” in *Handbook of Agricultural Economics*, ed. by B. Gardner, and G. Rausser, vol. 1, chap. 17, pp. 971–1024. Elsevier Science.
- FACKLER, P. L., AND H. TASTAN (2008): “Estimating the Degree of Market Integration,” *American Journal of Agricultural Economics*, 90(1), 69–85.
- GALLANT, A., D. HSIEH, AND G. TAUCHEN (1997): “Estimation of Stochastic Volatility Models with Diagnostics,” *Journal of Econometrics*, 81, 159–192.

- GALLANT, A., AND D. NYCHKA (1987): "Semi-nonparametric Maximum Likelihood Estimation," *Econometrica*, 55, 363–390.
- GALLANT, A., AND G. TAUCHEN (1989): "Semi-nonparametric Estimation of Conditionally Constrained Heterogenous Processes: Asset Pricing Applications," *Econometrica*, 57, 1091–1120.
- (1996): "Which Moments to Match," *Econometric Theory*, 12, 657–681.
- GHYSELS, E., AND A. GUAY (2003): "Structural Change Tests for Simulated Method of Moments," *Journal of Econometrics*, 115, 91–123.
- (2004): "Testing for Structural Change in the Presence of Auxiliary Models," *Econometric Theory*, 20, 1168–1202.
- GHYSELS, E., L. KHALAF, AND C. VODONOU (1994): "Simulation Based Inference in Moving Average Models," CIRANO working paper series, Montreal.
- GOURIEROUX, C., AND A. MONFORT (1996): *Simulation-Based Econometric Methods*. Oxford University Press, New York, 1st edn.
- GOURIEROUX, C., A. MONFORT, AND E. RENAULT (1993): "Indirect Inference," *Journal of Applied Econometrics*, 8, S85–S118.
- HALL, A. (2005): *Generalized Method of Moments*, Advanced Texts in Econometrics. Oxford University Press, New York, 1st edn.
- JONES, S. R., C. S. PERTTUNEN, AND B. E. STUCKMAN (1993): "Lipschitzian Optimization Without the Lipschitz Constant," *Journal of Optimization Theory and Applications*, 79, 157–181.
- JUDD, K. L. (1998): *Numerical Methods in Economics*. The MIT Press, Cambridge, MA, 1st edn.
- LEE, B., AND B. F. INGRAM (1991): "Simulation Estimation of Time Series Models," *Journal of Econometrics*, 47, 197–205.
- LIU, M. (2000): "Modeling Long Memory in Stock Market Volatility," *Journal of Econometrics*, 99, 139–171.
- LONG, A. R. G. J. R. (1997): "Estimating Stochastic Differential Equations Efficiently by Minimum Chi-squared," *Biometrika*, 84(1), 125–141.
- MARTIN, V. L., AND N. P. WILKINS (1999): "Indirect Estimation of ARFIMA and VARFIMA Models," *Journal of Econometrics*, 93, 149–175.
- McFADDEN, D. (1989): "A Method of Simulated Moments for Estimation of Discrete Response Models Without Numerical Integration," *Econometrica*, 57(5), 995–1026.
- MCNEW, K., AND P. FACKLER (1997): "Testing Market Equilibrium: Is Cointegration Informative," *Journal of Agricultural and Resource Economics*, 22(2), 191–207.

- MICHAELIDES, A., AND S. NG (2000): “Estimating the Rational Expectations Model of Speculative Storage: A Monte Carlo Comparison of Three Simulation Estimators,” *Journal of Econometrics*, 96, 231–266.
- MIRANDA, M. J., AND P. FACKLER (2002): *Applied Computational Economics and Finance*. MIT Press, 1st edn.
- MONFARDINI, C. (1998): “Estimating Stochastic Volatility Model through Indirect Inference,” *Econometrics Journal*, 1, C113–C128.
- NEWBY, W. K., AND K. D. WEST (1987): “A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix,” *Econometrica*, 55(3), 703–708.
- PAKES, A., AND D. POLLARD (1989): “Simulation and the Asymptotics of Optimization Estimators,” *Econometrica*, 57(5), 1027.
- SMITH, A. A. (1993): “Estimating Nonlinear Time Series Models Using Simulated Vector Autoregressions,” *Journal of Applied Econometrics*, 8, S63–S84.
- TASTAN, H. (2003): “Simulation-Based Estimation of Spatial Price Equilibrium Models and Market Integration,” Ph.D. thesis, North Carolina State University, Raleigh, NC, USA.
- VAN DER SLUIS, P. J. (1997): “Emm Pack 1.01: C/C++ Code for use with Ox for Estimation of Univariate Stochastic Volatility Models with the Efficient Method of Moments,” *Studies in Nonlinear Dynamics and Econometrics*, 2(3), 77–94.

Appendix

This appendix describes three “generic” moment functions that are provided with the MATLAB code available on the first author’s web site. These functions are useful on their own and also provide additional examples of how the moment functions are coded. The first is an additively separable model which uses the first k moments and cross moments of a set of d variables. For example, if $d = 2$ and $k = 3$ the moment function would return the sample mean of

$$[y_1 \ y_2 \ y_1^2 \ y_1 y_2 \ y_2^2 \ y_1^3 \ y_1^2 y_2 \ y_1 y_2^2 \ y_2^3]$$

The code for this moment function is

```
function out=allmom(beta,y,avg,k);
m=crossmom(y,k);
if isempty(beta)
    out=mean(m)';
else
    beta=beta';
    out=m-beta(ones(size(y,1),1),:);
end
```

The main work is done by the utility function `crossprod` which has the syntax

```
m=crossmom(y,k);
```

When passed an $n \times d$ matrix `y` and a nonnegative integer `k`, `crossmom` returns an n -row matrix containing all of the cross products of `y` up to order k (`crossmom` is coded in C for increased efficiency).⁷

The second “generic” moment function, `varmom`, returns either scores or the parameters of a VAR(k) model. The function `varmom` can be useful when the dynamics of the structural model can be summarized by a VAR model. The calling syntax is

```
function out=varmom(beta,y,avg,k)
```

where `beta` is the auxiliary parameter vector, `y` is either simulated or observed series and `k` is the VAR lag length. If `beta` is input as an empty vector `varmom` returns VAR parameters, otherwise it returns score vector of VAR(k) process. If `avg` is set to 0 `varmom` returns individual scores otherwise it returns their mean.

The third “generic” moment function computes the score function for the Semi-Nonparametric (SNP) density (Gallant and Nychka). This density for a d -dimensional random variable y is given by

$$f(y) = \frac{(P(z, k)\alpha)^2 \phi(z)}{\|L\|}$$

⁷The number of cross product terms is $(k + d)! / (k!d!) - 1$.

where $z_t = L^{-1}(y - \mu)$, ϕ is the standard normal density and $P(z, k)$ is a complete set of modified Hermite polynomials of z up to order k (there are a total of $(k + d)!/(k!d!)$ columns).⁸

The parameters of the model to be estimated are μ ($d \times 1$), $\text{vech}(L)$, where L is an lower triangular $d \times d$ matrix and the α_j (the code is actually written to impose the normalization that $\sum_i \alpha_i^2 = 1$). A separate document discussing estimation using the SNP distribution is available from the first author. The syntax for the function is

```
[score,dscore]=
  snpmom(beta,y,avg,k,mu,L,eta,alpha,dalpha,alphahat,alphatilde,Phi)
```

or

```
[score,dscore]= snpmom(beta,y,avg,mp{:})
```

where k is the order of the SNP density and `mp` is a cell array containing additional inputs which can be extracted using the function `snmpmp` (see below). To use the scores of SNP density as the moment function, users must first call the `snpmle` function to obtain the ML estimates. The syntax is

```
[beta,mu,L,eta]=snpmle(y,k,options);
```

Users can specify optimization parameters using `options` (see documentation). The toolbox uses divided rectangles global optimization (DiRect) algorithm developed by Jones, Perttunen, and Stuckman (1993) to explore the surface of the SNP log-likelihood function within initial parameter bounds. Then, parameter estimates from the DiRect algorithm are used as the starting values for the quasi-Newton algorithm. The function `snmpmp` extracts other parameters to pass to the `snpmom` function:

```
mp=snmpmp(beta,d,k);
```

where output is a cell array:

```
mp={k,mu,L,eta,alpha,dalpha,alphahat,alphatilde,Phi}
```

The function `snmpmp` calls `snpextract` to extract the additional parameters

```
[mu,L,eta,alpha,dalpha,alphahat,alphatilde,Phi,Psi] =
  snpextract(beta,d,k,Phi,Psi);
```

where `d` is the dimension of y (`d=size(y,2)`). The weights, `Phi` and `Psi` can be obtained using

⁸More general versions of the SNP model that allow for dynamic effects and for conditioning on exogenous variables are possible but are not currently implemented in this code library.

```
[Phi,Psi]=snweights(d,k);
```

A call to snpmom with the following syntax

```
m=snpmom(beta,y,0,mp{:});
```

returns individual scores.