Inference Using Simulated Neural Moments (in progress draft, do not cite)

Michael Creel*

September 18, 2020

Abstract

This paper deals with Laplace type methods used with moment-based, simulation-based, econometric estimators. It shows that confidence intervals based upon quantiles of a tuned MCMC chain may have coverage which is far from the nominal level. It discusses how neural networks may be used to easily and automatically reduce the dimension of an initial set of moments to the minimum number of moments needed to maintain identification. When estimation and inference is based on the neural moments, which are the result of filtering moments through a trained neural net, confidence intervals have correct coverage in almost all cases, and departures from correct coverage are small.

Keywords: neural networks; Laplace type estimators; simulation-based estimation
JEL codes: C11, C12, C13, C45

1 Introduction

It has long been known that classical inference methods based on first-order asymptotic theory, when applied to the generalized method of moments estimator, may lead to unreliable results, in the form of substantial finite sample biases and variances, and incorrect coverage of confidence intervals, especially when the model is overidentified (Tauchen (1986), Hall and Horowitz (1996), Hansen et al. (1996), Donald et al. (2009)). Chernozhukov and Hong (2003) introduced Laplace type estimators, which allow for estimation and inference with classical statistical methods (those which are defined by optimization of an objective function) to be done by working with the elements of a tuned Markov chain, so that potentially difficult or unreliable steps such as optimization or computation of asymptotic standard errors, etc., may be avoided. Another important strand of literature is simulation-based estimation. The strands of moment-based estimation, simulation, and Laplace type methods meet in certain applications. The code by Gallant and Tauchen (Gallant and Tauchen (2002)) for efficient method of moments estimation (Gallant and Tauchen (1996)), which has been used in numerous papers, is an example. Another is Christiano et al. (2010), which proposes a Laplace type estimation methodology that uses simulated moments which are defined in terms of impulse response functions, for estimation of macroeconomic models. Similar methodologies may be found in the very broad Approximate Bayesian Computing literature, some of which uses MCMC methods and criterion functions that involve simulated moments (*e.g.,* Marjoram et al. (2003)).

Given the uneven performance of inference in classical GMM applications, one may wonder how reliable are inferences made using the combination of Laplace type methods and simulated moments. This paper provides evidence that confidence intervals may have poor coverage in some cases, and it provides evidence that a dimension reduction technique based computing simulated moments using a trained neural net can cause inferences to become much more reliable. The results are for three simple simulation examples, and are, thus, by no means conclusive. However, a software framework is provided that will allow the proposed methods to be easily applied to estimation of other simulable models.

*Universitat Autònoma de Barcelona, Barcelona GSE and MOVE. michael.creel@uab.cat. The financial support from the Spanish Ministry of Science, Innovation and Universities and FEDER through grant PGC2018-094364-B-I00 is gratefully acknowledged.

The next section reviews how Laplace type methods may be used with simulated moments, and Section 3 then discusses how neural networks may be used to reduce the dimension of the moment conditions. Section 4 gives results for three simple models, and a final section summarizes the conclusions.

2 Simulated Moments, Indirect Likelihood, and Laplace Type Inference

This section relies on results from the part of the simulation-based estimation literature which bases estimation on a statistic, including McFadden (1989), Gouriéroux et al. (1993), Smith (1993) and Gallant and Tauchen (1996), among others, which is reviewed in Jiang and Turnbull (2004)). Suppose there is a model $M(\theta)$ which generates data from a probability distribution $P^{(\theta)}$ which depends on the unknown parameter vector θ. $M(\theta)$ is fully known up to θ, so that we can make draws of the data from the model, given θ. Let $Y = Y(\theta)$ be a sample drawn at the parameter vector θ, where $\theta \in \Theta \subset \mathbb{R}^k$ and Θ is a known parameter space. Suppose we have selected a finite dimensional statistic $Z(\theta) = Z(Y(\theta))$ upon which to base estimation, and assume that the statistic satisfies a central limit theorem, uniformly, for all values of θ of interest:

$$\sqrt{n}\,(Z - E_\theta Z) \to^d N(0, \bar{\Sigma}(\theta)) \tag{1}$$

Let $Z^s(\theta) = Z(Y^s(\theta))$ be the statistic evaluated using an artificial sample drawn from the model at the parameter value θ. This statistic has the same asymptotic distribution as does $Z(\theta)$, and furthermore, the two statistics are independent of one another. With S such simulated statistics, define $m(\theta) = Z(\theta) - S^{-1}\sum_s Z^s(\theta)$ and $\bar{V}(\theta) = (1 + S^{-1})\bar{\Sigma}(\theta)$. We can easily obtain

$$\sqrt{n}m(\theta) \to^d N(0, \bar{V}(\theta)). \tag{2}$$

Now, suppose we have a real sample which was generated at the unknown true parameter value θ_0, and let $\hat{Z}$ be the associated value of the statistic. Define $\hat{m}(\theta) = \hat{Z} - S^{-1}\sum_s Z^s(\theta)$. With this, and eqn. 2,we can define the indirect likelihood function[1]

$$L = L(\theta|\hat{Z}) = \left|2\pi\hat{\bar{V}}(\theta)\right|^{-1/2} \exp(-\frac{1}{2}H) \tag{3}$$

where

$$H = H(\theta|\hat{Z}) = n \cdot \hat{m}(\theta)^T \hat{\bar{V}}^{-1}(\theta)\hat{m}(\theta). \tag{4}$$

where $\hat{\bar{V}}(\theta)$ is a consistent estimate of $\bar{V}(\theta)$.

To estimate $\bar{V}(\theta)$, one possibility is to use a fixed sample-based estimate that does not rely on an estimate of θ_0 (see, for example, Christiano et al. (2010, 2016)). Another possibility is to compute the estimate $\hat{\bar{\Sigma}}(\theta)$ of the covariance matrix in 1 as the sample covariance of R draws of $\sqrt{n}Z^s(\theta)$, and then multiply the result by $1 + S^{-1}$ to obtain the estimate $\hat{\bar{V}}(\theta)$. This estimator may be used in a continuously updating fashion, by updating $\hat{\bar{V}}(\theta)$ in eqns. 3 or 4 every time the respective function is evaluated. Alternatively, if we obtain an initial consistent estimator of θ_0, then $\hat{\bar{V}}(\theta)$ can be computed at this estimate, and kept fixed in subsequent computations, in the usual two-step manner. Note that, if a fixed covariance estimator is used, then the maximizer of L is the same as the minimizer of H.

Extremum estimators may be obtained by maximizing $\log L$, or minimizing H. Laplace type estimators, as defined by Chernozhukov and Hong (2003), may be defined by setting their general criterion function, $L_n(\theta)$, as defined in their Section 3.1, to either $\log L$, or $-\frac{1}{2}H$. Once this is done, then the practical methodology is to use Markov chain Monte Carlo (MCMC) methods to draw a chain $C = \{\theta^r\}$, $r = 1, 2, ..., R$, given the sample statistic $\hat{Z}$, where acceptance/rejection is determined using the chosen $L_n(\theta)$, along with a prior, and standard proposal

[1] These definitions and notation are loosely based on Jiang and Turnbull (2004).

methods[2]. Thus, this paper will rely directly on the theory and methods of Chernozhukov and Hong (2003), just using the criterion functions presented above to define the specific Laplace type estimators. The chain may be initialized at the respective extremum estimator, $\hat{\theta} = \arg\max L_n(\theta)$, and this estimate may also be used to obtain an estimate of $\bar{V}(\theta)$, to use in a two-step version. As point estimators, we can use either the extremum estimator, or Bayesian point estimators such as the mean or median of the final chain. In the following, a primary use of the Chernozhukov and Hong (2003) methodology will be in order to obtain confidence intervals. For a function $f(\theta)$, Theorem 3 of Chernozhukov and Hong (2003) proves that a valid confidence interval can be obtained by using the quantiles of $\{f(\theta^r)\}_{r=1,2,..R}$, based on the final chain $C = \{\theta^r\}$, $r = 1, 2, ..., R$. For example, a 95% confidence interval for a parameter θ_j is given by the interval $(Q_{\theta_j}(0.025), Q_{\theta_j}(0.975))$, where $Q_{\theta_j}(\tau)$ is the τth quantile of the R values of the parameter θ_j in the chain C.

3 Neural Moments

The dimension of the statistics used for estimation, Z, can be made minimal (equal to the dimension of the parameter to estimate, θ) by filtering an initial set of statistics, say, W, through a trained neural net. Details of this process are explained in Creel (2017) and references cited therein, and the process is made explicit in the code archive which accompanies this paper[3]. A summary of this process is: Suppose that W is a p vector of statistics $W = W(Y)$, with $p \geq k$, where $k = \dim \theta$. We may generate a large sample of (W, θ) pairs, following:

1. draw θ^s from the parameter space Θ, using some prior distribution (*e.g.,* a uniform distribution over Θ).
2. draw a sample Y^s from the model $M(\theta)$ at θ^s
3. compute the vector of raw statistics $W(Y^s)$.

We can repeat this process to generate a large data set $\{\theta^s, W^s\}, s = 1, 2, ..., S$, which can be used to train a neural network which predicts θ, given W. This process can be done without knowledge of the real sample data, and can in fact be done before the real sample data is gathered. The prediction from the net will of the same dimension as θ, and if the net is of an appropriate configuration and is well-trained using a squared error loss function, the output of the net will be a very accurate approximation to the posterior mean of θ conditional on W. The output of the net may be represented as $\hat{\theta} = f(W, \hat{\phi})$,where $f(W, \phi) : R^p \to R^k$ is the neural net, with parameters ϕ, that takes as inputs the p statistics W,and has $k = \dim \theta$ outputs. The parameters of the net, ϕ, are adjusted using standard training methods from the neural net literature to obtain the trained parameters, $\hat{\phi}$. Then we can think of $\hat{\theta} = f(W, \hat{\phi})$ as a $k-$dimensional statistic which can be computed essentially instantaneously once provided with W. We will use this statistic $\hat{\theta}$ as the Z of the previous section. Because the statistic is an accurate approximation to the posterior mean conditional on W (supposing the net was well trained), it has two virtues: it is informative for θ (supposing that the initial statistics W contain information on θ) and it has the minimal dimension needed to identify θ. From the related GMM literature, GMM methods are known to lead to inaccurate inference when the dimension of the moments is large relative to the dimension of the parameter vector (Donald et al. (2009)). Use of a neural net as described here reduces the dimension of the statistic to the minimum required for identification.

When the statistic Z is the output of a neural net $f(W, \phi)$, where the parameter vector of the net, ϕ, can have a very high dimension (hundreds or thousands of parameters are not uncommon) the simulated likelihood of eqn. 3 will be a wavy function, with many local maxima. This will occur even if the net is trained using regularization methods. Because of this waviness, gradient-based methods will not be effective when attempting to maximize $\log L$ or to minimize H (eqns. 3 and 4), and attempts to compute the covariance matrix of the estimator that rely on derivatives of the log likelihood function will also fail. However, derivative free methods[4] can be used to compute extremum estimators, to obtain point estimators or to initialize a MCMC chain, and the simulation-based estimator of the covariance matrix $\bar{\Sigma}(\theta)$ of eqn. 1 discussed in the previous section does not depend on derivatives. It is worth noting that the output of the net evaluated at the real sample statistic, $f(\hat{W}, \hat{\phi})$, will also provide an excellent starting value for computing extremum estimators, or for initializing a MCMC chain.

[2]It may be noted that methods other than MCMC may be used to generate the set of draws from the posterior, C. For example, one might use sequential Monte Carlo. Point estimation and inference using C remains the same regardless of how C is generated.

[3]See https://github.com/mcreel/SNM and especially the file https://github.com/mcreel/SNM/blob/master/lib/Train.jl

[4]Simulated annealing (Goffe et al. (1994)) is used in what follows.

4 Examples

This section presents three simple examples to investigate the performance of the proposed methods. We begin by describing the models.

4.1 Models

For all models, the code used (for the Julia language[5]) is available in an archive[6], where the details of each example may be consulted. The three example models also serve as templates that may be used to apply to proposed methods to models of the reader's interest: one simply needs to provide similar functions to what is found in the directory for each example, for the model of interest. These are, fundamentally, 1) a prior from which to draw the parameters; 2) code to simulate the model given the parameter value, and finally, 3) code to compute the initial statistics, W, given the data generated from the model.

4.1.1 Stochastic Volatility

The simple stochastic volatility model is

$$\begin{aligned} y_t &= \phi \exp(h_t/2)\epsilon_t \\ h_t &= \rho h_{t-1} + \sigma u_t \end{aligned}$$

where ϵ_t and u_t are independent standard normal random variables. We use a sample size of 500 observations, and the true parameter values are $\theta_0 = (\phi_0, \rho_0, \sigma_0) = (0.692,\ 0.9,\ 0.363)$. These parameter values have been chosen to facilitate comparison with results of a number of previous studies that have used the SV model to check properties of estimators.

For estimation, 11 statistics are used to form the initial set, W, which include moments of y and of $|y|$, as well as the estimated parameters of a HAR auxiliary model (Corsi (2009)) fit to $|y|$.[7]

4.1.2 Dynamic Panel Data

The dynamic panel data model is borrowed from Forneron and Ng (2018), who adapted the model of Gouriéroux et al. (2010). The model is

$$y_{it} = \alpha_i + \rho y_{it-1} + \beta x_{it} + \sigma \epsilon_{it}$$

where α_i, x_{it}, and ϵ_{it} are all mutually independent standard normal random variables, for $i = 1, 2, ..., n$, and $t = 1, 2, ..., T$. We set $T = 5$ and $n = 100$. For initialization, y_{i0} is generated as a draw from it's unconditional distribution, for each i. We estimate the parameter vector $\theta_0 = (\rho_0, \beta_0, \sigma_0^2)$, where the true values are 0.6, 1.0 and 2.0, respectively. This is the same design as is used by Forneron and Ng (2018) in their Table 3, to facilitate comparison.

Eight statistics are included in the initial set, W, for estimation of the three parameters. The first four are the three ordinary least squares estimates of the regression $y_{it} = \phi_1 + \phi_2 y_{it-1} + \phi_3 x_{it} + \eta_{it}$, which simply ignores the panel data structure, along with the estimated variance of the error term. The next four statistics are the fixed effects estimator, obtained by subtracting cross sectional means from all variables.

4.1.3 ARMA

The next example is a simple ARMA(1,1) model

$$\begin{aligned} x_t &= \alpha x_{t-1} + f_t - \beta f_{t-1} \\ f_t &\sim IIN(0, \sigma^2), \end{aligned}$$

[5] https://julialang.org/
[6] https://github.com/mcreel/SNM
[7] See the file https://github.com/mcreel/SNM/blob/master/SV/SVlib.jl for details.

with true values $\theta_0 = (\alpha_0, \beta_0, \sigma_0^2) = (0.95,\ 0.5,\ 1.0)$. The sample size is $n = 300$. This is the same design as is used by Fiorentini et al. (2018), in their Table 1 (middle section), in work that studies, in part, indirect inference methods.

The 13 statistics used to define the initial set, W, include sample moments and correlations, OLS estimates of an AR(1) auxiliary model fit to x_t, as well as an another AR(1) model fit to the residuals of the first model, plus partial autocorrelations of x_t[8]

4.1.4 Mixture of Normals

The final example model is a mixture of normals. The variable y is drawn from the distribution $N(\mu_1, \sigma_1^2)$ with probability p and from $N(\mu_2, \sigma_2^2)$ with probability $1 - p$. Samples of 1000 observations are drawn. The true parameter values are $\theta_0 = (\mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1.0,\ 0.2,\ 0.0,\ 2.0,\ 0.4)$. The 11 auxiliary statistics are the sample minimum and maximum, and the 0.1, 0.2,...,0.9 quantiles.

5 Results

5.1 Continuous updating

This section report results for Laplace type MCMC estimation of each of the example models, using the GMM-like criterion function H (see eqn. 4) as the L_n of Chernozhukov and Hong (2003). Two versions are reported: first, using the initial statistics, W, and, second, using the statistics Z which are the output of the trained neural net. Results using the criterion L (see eqn. 3) are qualitatively very similar, and are thus omitted. Results include root mean squared error (RMSE) and bias for the extremum estimator which minimizes H, as this was found to have slightly better (but qualitatively very similar) performance than the posterior mean or median of the tuned MCMC chain, and coverage of 90, 95 and 99% confidence intervals computed using the appropriate quantiles of the final tuned MCMC chain, following Chernozhukov and Hong (2003). In all cases, 500 Monte Carlo replications were used. The covariance matrix estimator $\hat{\hat{V}}(\theta)$ was continuously updated at every evaluation of the criterion.

Table 1 reports RMSE. RMSE is in most cases not dramatically different between estimators that use the full set of statistics and those based on the neural net filtered statistics, and there is no clear pattern in the differences that do exist. In six of nine cases, the neural net filtered statistics lead to lower RMSE, but in most instances, the difference is fairly small. There is no clear reason to prefer one version based on RMSE.

Table 2 reports bias. In all cases, bias is low, and there is no clear pattern of differences between estimators based on W and those based on Z.

Results for confidence interval coverage are presented in Tables 3, 4 and 5, for 90, 95 and 99 percent intervals, respectively. Monte Carlo confidence interval coverage is the proportion of times that the true parameter value is not rejected, at the chosen confidence level. Before interpreting results, we remind the reader that 500 Monte Carlo replications were done in all cases. Critical values for the hypothesis that a $100 \cdot p\%$ confidence interval has correct coverage can be found using the quantiles of a binomial(500,p) random variable. In Tables 3-5, cases where correct coverage is rejected at the 1% significance level are indicated by italic typeface. Here, we see some important differences. Coverage is poor for estimators that directly use the full set of moments, W, for the SV and ARMA models, where correct coverage is rejected in all cases except for one of 18 (the 90% interval for the ϕ parameter of the SV model). For the DPD model, the estimators based on W and on Z both perform well, as correct coverage is never rejected. For the three models, when the neural net statistics are used, correct coverage is not rejected except for 2 of 18 cases, and in both of these cases, the error is one of Monte Carlo Type I error being lower than the nominal level, which is to say that the confidence intervals are somewhat broader than they should be, erring on the side of caution.

[8] Details are in the file https://github.com/mcreel/SNM/blob/master/ARMA/ARMAlib.jl.

5.2 Two step estimator

In order to speed up computations, a possibility is to use a consistent estimator of the parameter vector to compute an estimate of the covariance matrix $\bar{V}(\theta)$, and to keep this estimate fixed when evaluating the criterion function during the course of MCMC iterations, in typical two-step fashion. Concretely, the parameter vector θ was consistently estimated using $\arg\min H(\theta)$, where $\hat{\bar{V}}(\theta)$ in eqn. 4 was set to an identity matrix, using simulated annealing to do the minimization. Then $\hat{\bar{V}}(\theta)$ was fixed to its value at this estimate of the parameter vector, and MCMC was done in the same fashion as in the previous section, with this one difference. This makes the MCMC iterations faster, as the simulation-based update of the covariance estimate at each MCMC step is avoided. This estimation method was applied to the Mixture of Normals model. In this case, 1000 Monte Carlo replications were done, and only the neural statistics were used, as the previous results support the use of these, in favor of the whole set of statistics. We see in Tables 3-5 that confidence interval coverage is statistically proper in all occasion except one (the 99% interval for the σ_1 parameter). Thus, the experiment supports the use of the two step version of the methods, which is of interest because it is less computationally intensive.

6 Application: A Jump Diffusion Model of S&P 500 Returns

The previous examples are all small models that are not costly to simulate. As an example of a more computationally challenging model, this section presents results for estimation of a jump diffusion model of S&P 500 returns. Solving and simulating[9] the model for each MCMC trial parameter acceptance/rejection decision takes about 13 seconds, so training a net and estimation by MCMC is somewhat costly, requiring about 2 days to complete using a moderate power workstation and threads-based parallelization, where possible. This example is intended to show that the methods are feasible for research projects where simulation from the model is costly, but not extremely so.

The jump diffusion model is

$$
\begin{aligned}
dp_t &= \mu dt + \sqrt{\exp h_t} dW_{1t} + J_t dN_t \\
dh_t &= \kappa(\alpha - h_t) + \sigma dW_{2t}
\end{aligned}
$$

where p_t is log price, h_t is log volatility, J_t is jump size, and N_t is a Poisson process with jump intensity λ_0. W_{1t} and W_{2t} are two standard Brownian motions with correlation ρ. When a jump occurs, its size is $J_t = a\lambda_1\sqrt{\exp h_t}$, where a is 1 with probability 0.5 and -1 with probability 0.5. So, jump size depends on the current standard deviation, and jumps are positive or negative with equal probability. Log price, p_t, is simulated using 5 minute tics, and the observed log price adds a $N(0, \tau^2)$ measurement error to p_t. From this model, 1000 daily observations on returns, realized volatility (RV), and bipower variation (BV) are generated. The model is simulated on a continuous 24 hour basis, and returns are computed using the change in daily log closing price, for trading days only. Overnight periods and weekends are simulated, but returns, RV and BV are recorded only at the close of trading days. In summary, the seven parameters are $\theta = (\mu, \kappa, \alpha, \sigma, \rho, \lambda_0, \lambda_1, \tau)$, and simulated data consists of 1000 daily observations on returns, RV and BV.

The raw statistics, W, which are used to train the net and to do estimation are a combination of coefficients from auxiliary regressions between the three observed variables, summary statistics, and functions of quantiles of the variables. It is possible to tune the choice of statistics through experimentation with artificially generated data, to ensure that the parameters are all well identified, and it is possible to check that statistics have a minimal importance, by examining the weights of the first layer of the neural net[10]. The details of the 25 statistics which are used are found in the file JDlib.jl. The importances of the statistics are seen in the figure ImportanceOfStatistics.svg.

The model was fit to S&P500 data[11] from 16 Dec. 2013 to 04 Dec. 2017, which is an interval of 1000 trading days, the same as was used to train the neural net. ADD PLOT OF data. The results are in Figures x-y, which

[9] The model is solved and simulated using the SRIW1 strong order 1.5 solver from the DifferentialEquations.jl package for the Julia language.

[10] See the file Importance.jl for how to do this

[11] The data source is the Oxford-Man Institute's realized library, v. 0.2.

show nonparametric plots of the marginal posterior density for each parameter, along with posterior means and medians, and 90% confidence intervals defined by the limits of the green areas.

7 Conclusions

This paper has shown that confidence intervals based upon quantiles of a tuned MCMC chain may have coverage which is far from the nominal level. It has proposed to use neural networks to reduce the dimension of an initial set of moments to the minimum number of moments needed to maintain identification. When estimation and inference is based on the neural moments, confidence intervals have correct coverage in almost all cases, and departures from correct coverage are small.

It is to be noted that the step of filtering moments though a neural net is very easy and quick to perform using modern deep learning software environments. The software archive that accompanies this paper provides a function for automatic training, requiring no human intervention. It only requires a sample of moments drawn from the model at parameter values drawn from the prior, and this is always possible when simulation-based estimation is being used. Filtering moments through a neural net gives an informative, minimal dimension statistic as the output. This provides an alternative to moment selection procedures. Uninformative moments are essentially removed, and correlated moments are combined.

This paper has examined how inference using quantiles of MCMC chains may be improved when neural moments are used. It seems likely that other inference methods which are used with simulation-based estimators, such as bootstrapping, may be made more reliable if neural moments are used, as dimension reduction while maintaining relevant information is likely to be generally beneficial.

References

Victor Chernozhukov and Han Hong. An MCMC approach to classical estimation. *Journal of Econometrics*, 115(2): 293–346, aug 2003. doi: 10.1016/s0304-4076(03)00100-3. URL http://dx.doi.org/10.1016/s0304-4076(03)00100-3.

Lawrence J Christiano, Mathias Trabandt, and Karl Walentin. Dsge models for monetary policy analysis. In *Handbook of Monetary Economics*, volume 3, pages 285–367. Elsevier, 2010.

Lawrence J Christiano, Martin S Eichenbaum, and Mathias Trabandt. Unemployment and business cycles. *Econometrica*, 84(4):1523–1569, 2016.

Fulvio Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196, 2009.

Michael Creel. Neural nets for indirect inference. *Econometrics and Statistics*, 2:36–49, 2017. URL https://doi.org/10.1016/j.ecosta.2016.11.008.

Stephen G. Donald, Guido W. Imbens, and Whitney K. Newey. Choosing instrumental variables in conditional moment restriction models. *Journal of Econometrics*, 152(1):28–36, 2009. ISSN 0304-4076. doi: 10.1016/j.jeconom.2008.10.013. URL http://www.sciencedirect.com/science/article/pii/S0304407609000566. Recent Adavances in Nonparametric and Semiparametric Econometrics: A Volume Honouring Peter M. Robinson.

Gabriele Fiorentini, Alessandro Galesi, and Enrique Sentana. A spectral em algorithm for dynamic factor models. *Journal of Econometrics*, 205(1):249–279, 2018.

Jean-Jacques Forneron and Serena Ng. The abc of simulation estimation with auxiliary statistics. *Journal of Econometrics*, 205(1):112–139, 2018.

A Ronald Gallant and George Tauchen. Emm: A program for efficient method of moments estimation, version 1.6, user's guide. *Manuscript, University of North Carolina*, 2002.

R. Gallant and G. Tauchen. Which moments to match? *Econometric Theory*, 12:363–390, 1996. doi: 10.2139/ssrn.37760. URL http://dx.doi.org/10.2139/ssrn.37760.

William L Goffe, Gary D Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60(1-2):65–99, 1994.

C. Gouriéroux, A. Monfort, and E. Renault. Indirect inference. *Journal of Applied Econometrics*, pages S85–S118, 1993. doi: 10.1002/jae.3950080507. URL http://dx.doi.org/10.1002/jae.3950080507.

Christian Gouriéroux, Peter CB Phillips, and Jun Yu. Indirect inference for dynamic panel models. *Journal of Econometrics*, 157(1):68–77, 2010.

Peter Hall and Joel L Horowitz. Bootstrap critical values for tests based on generalized-method-of-moments estimators. *Econometrica*, pages 891–916, 1996.

Lars Peter Hansen, John Heaton, and Amir Yaron. Finite-sample properties of some alternative GMM estimators. *Journal of Business & Economic Statistics*, 14(3):262–280, jul 1996. doi: 10.1080/07350015.1996.10524656. URL http://dx.doi.org/10.1080/07350015.1996.10524656.

Wenxin Jiang and Bruce Turnbull. The indirect method: inference based on intermediate statistics a synthesis and examples. *Statistical Science*, 19(2):239–263, 2004.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

Daniel McFadden. A method of simulated moments for estimation of discrete response models without numerical integration. *Econometrica*, 57(5):995–1026, 1989. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913621.

Anthony A Smith. Estimating nonlinear time-series models using simulated vector autoregressions. *Journal of Applied Econometrics*, 8(S1), 1993.

George Tauchen. Statistical properties of generalized method-of-moments estimators of structural parameters obtained from financial market data. *Journal of Business & Economic Statistics*, 4(4):397, oct 1986. doi: 10.2307/1391493. URL http://dx.doi.org/10.2307/1391493.

Tables

Table 1: RMSE of $\arg\min H$,using raw (W) or neural net (Z) statistics

Model	Parameter	true value	W	Z
SV	ϕ	0.692	0.123	0.075
	ρ	0.90	0.086	0.072
	σ	0.363	0.138	0.133
DPD	ρ	0.6	0.036	0.034
	β	1.0	0.075	0.067
	σ^2	2.0	0.139	0.151
ARMA	α	0.95	0.027	0.043
	β	0.5	0.079	0.081
	σ^2	1.0	0.099	0.086

Table 2: Bias of $\arg\min H$,using raw (W) or neural net (Z) statistics

Model	Parameter	true value	W	Z
SV	ϕ	0.692	0.011	-0.018
	ρ	0.90	0.001	0.003
	σ	0.363	0.012	-0.006
DPD	ρ	0.6	-0.001	0.002
	β	1.0	0.000	0.000
	σ^2	2.0	0.010	0.003
ARMA	α	0.95	-0.004	-0.005
	β	0.5	0.008	0.001
	σ^2	1.0	0.004	-0.006

Table 3: 90% confidence interval coverage using H to define the Laplace type estimator, using raw (W) or neural net (Z) statistics. Italic typeface indicates that correct coverage is rejected at the 1% level.

Model	Parameter	true value	W	Z
SV	ϕ	0.692	0.876	*0.936*
	ρ	0.90	*0.732*	0.894
	σ	0.363	*0.762*	0.88
DPD	ρ	0.6	0.888	0.896
	β	1.0	0.900	0.914
	σ^2	2.0	0.908	0.898
ARMA	α	0.95	*0.786*	0.914
	β	0.5	*0.814*	*0.938*
	σ^2	1.0	*0.808*	0.908
MN	μ_1	1.0	na	0.918
	σ_1	0.2	na	0.875
	μ_2	0.0	na	0.908
	σ_2	2.0	na	0.908
	p	0.4	na	0.926

Table 4: 95% confidence interval coverage using H to define the Laplace type estimator, using raw (W) or neural net (Z) statistics. Italic typeface indicates that correct coverage is rejected at the 1% level.

Model	Parameter	true value	W	Z
SV	ϕ	0.692	*0.916*	0.966
	ρ	0.90	*0.796*	0.950
	σ	0.363	*0.824*	0.950
DPD	ρ	0.6	0.966	0.956
	β	1.0	0.966	0.960
	σ^2	2.0	0.954	0.946
ARMA	α	0.95	*0.838*	0.966
	β	0.5	*0.856*	0.966
	σ^2	1.0	*0.880*	0.954
MN	μ_1	1.0	na	0.965
	σ_1	0.2	na	0.931
	μ_2	0.0	na	9.957
	σ_2	2.0	na	0.955
	p	0.4	na	0.962

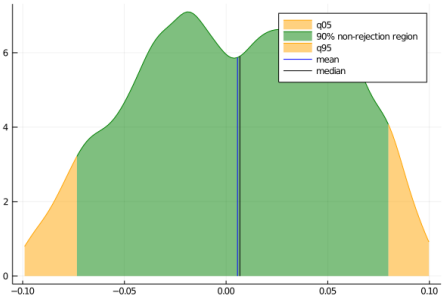
Table 5: 99% confidence interval coverage using H to define the Laplace type estimator, using raw (W) or neural net (Z) statistics. Italic typeface indicates that correct coverage is rejected at the 1% level.

Model	Parameter	true value	W	Z
SV	ϕ	0.692	*0.936*	0.984
	ρ	0.90	*0.848*	0.988
	σ	0.363	*0.888*	0.984
DPD	ρ	0.6	0.996	0.994
	β	1.0	0.988	0.986
	σ^2	2.0	0.996	0.978
ARMA	α	0.95	*0.898*	0.998
	β	0.5	*0.916*	0.988
	σ^2	1.0	*0.920*	0.992
MN	μ_1	1.0	na	0.992
	σ_1	0.2	na	*0.977*
	μ_2	0.0	na	0.990
	σ_2	2.0	na	0.986
	p	0.4	na	0.993

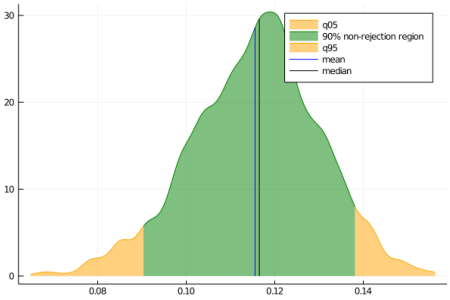
Figures

Figure 1: MCMC results for the S&P 500 data.

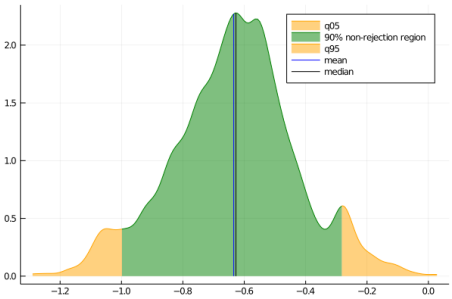
(a) μ



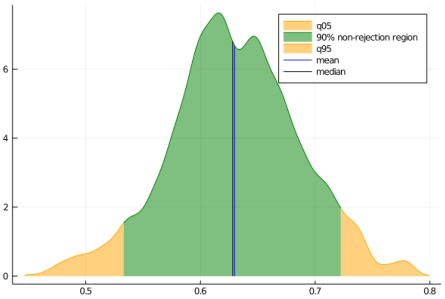
(b) κ



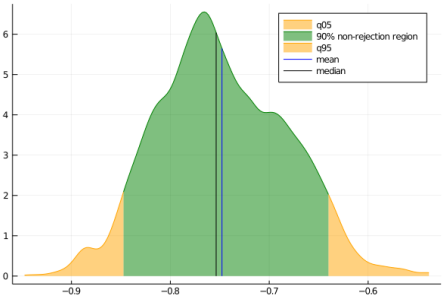
(c) α



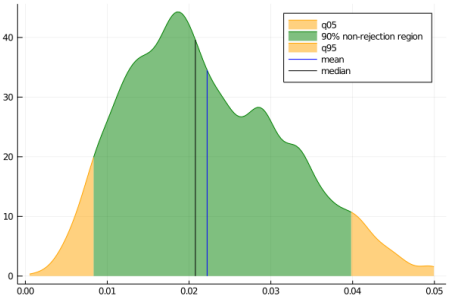
(d) σ



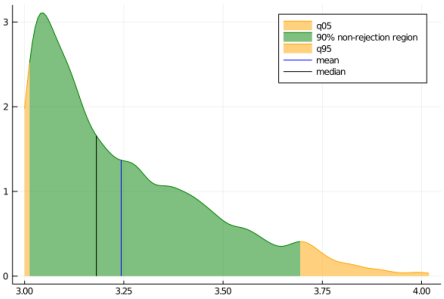
(e) ρ



(f) λ_0



(g) λ_1



(h) τ